Programmation en nombres entiers pour les diagrammes d'influence et les problèmes de maintenance d'une compagnie aérienne

Victor Cohen

UNIVERSITÉ PARIS-EST

École doctorale MATHÉMATIQUES ET SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Spécialité : Mathématiques

Présentée par

Victor COHEN

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-EST

INTEGER PROGRAMMING FOR INFLUENCE DIAGRAMS AND AIRLINE MAINTENANCE PROBLEMS

PROGRAMMATION EN NOMBRES ENTIERS POUR LES DIAGRAMMES D'INFLUENCE ET LES PROBLÈMES DE MAINTENANCE D'UNE COMPAGNIE AÉRIENNE

Soutenance le 18 décembre 2020 devant le jury composé de :

Patrick JAILLET	*MIT*	Rapporteur
Andrea LODI	*Ecole Polytechnique de Montréal*	Rapporteur
Georgina HALL	*INSEAD*	Examinatrice
Guillaume OBOZINSKI	*EPFL*	Examinateur
Vianney PERCHET	*ENSAE*	Examinateur
Frédéric MEUNIER	*École des Ponts ParisTech*	Directeur de thèse
Axel PARMENTIER	*École des Ponts ParisTech*	Encadrant de thèse

À mes parents et mes frères,

À Gabrielle.

Remerciements

Il est difficile d'écrire des remerciements pour tant de personnes qui ont compté dans mon épanouissement personnel et professionnel durant ces trois belles années de thèse. J'ai appris de tous ces échanges avec vous, qu'ils soient dans le cadre de la thèse ou en dehors, et j'espère n'oublier personne dans cette tentative.

En premier lieu, je tiens à remercier mes directeurs de thèse, Frédéric Meunier et Axel Parmentier. Merci à toi Frédéric pour m'avoir donné l'envie de faire une thèse et pour ton suivi rigoureux, notamment pendant la rédaction du manuscrit en plein confinement. Nos discussions étaient toujours enrichissantes pour moi grâce à tes connaissances et la passion que tu transmets. Aussi c'était un réel plaisir d'avoir avec toi de réjouissantes digressions sur l'histoire des mathématiques ou sur le jazz, plus particulièrement la discographie de Miles Davis. Merci à toi Axel pour m'avoir si bien accompagné quotidiennement durant ma thèse. Je souhaite à tous les doctorants d'avoir un directeur de thèse comme toi. Travailler avec toi est un vrai bonheur, notamment grâce à ton enthousiaste, ta disponibilité et ta curiosité. Les après-midi devant un tableau à chercher des preuves me manqueront, tout comme nos déjeuners à discuter de voyages et randonnées. J'ajoute à cela un excellent souvenir de notre voyage à Tokyo, avec notamment un fou rire mémorable dans un restaurant.

I thank Andrea Lodi and Patrick Jaillet for being the referees of my thesis and for their helpful remarks. Merci à toi Patrick pour m'avoir accueilli au MIT pendant quelques semaines qui ont été très enrichissantes pour moi. Je souhaite également remercier chaleureusement les examinateurs qui ont accepté de faire partie de mon jury : Georgina Hall, Vianney Perchet et Guillaume Obozinski.

Je remercie particulièrement Guillaume pour ses précieux conseils sur l'après-thèse. Durant ma thèse, j'ai eu l'opportunité de collaborer sur un projet avec Axel, Guillaume, ainsi que Vincent Leclère et Joseph Salmon. Je vous remercie tous pour m'avoir accueilli sur ce beau projet, j'ai beaucoup appris à vos côtés.

Je tiens à remercier les membres du département de Recherche Opérationnelle d'Air France, avec qui j'ai eu le plaisir de travailler ou d'échanger tout au long de ma thèse. Je remercie particulièrement Paul Louis Vincenti qui a appuyé le projet après mon stage et Solène Richard qui a suivi ma thèse et mis en avant les résultats auprès du reste du département de RO et des responsables de la maintenance. Merci à Laurent Demeestere, Jules Humbert, Robin Dupont et Patrick Marshall pour toute leur aide, notamment dans l'utilisation de Spark et Hadoop sur le cluster. Je remercie aussi Jules Dubois et Léo Pallud pour nos discussions sur l'ordonnancement des tâches de maintenance, ainsi que tous les membres des l'équipe de maintenance prédictive

avec qui j'ai eu l'occasion d'interagir. Je souhaite remercier les responsables du département Julie Pozzi, Benoit Robillard et Marine Le Touzé. Merci à tous les membres du département avec qui j'ai eu le plaisir de prendre un café, prendre une bière ou jouer au foot, et en particulier Pierre, Blaise, Alexandre, Guillaume, Benoît, Valentina, Magdalena, Ferran, Kevin, et tout ceux que je m'excuse d'oublier.

Merci à tous les chercheurs du Cermics avec qui j'ai eu le plaisir de discuter. Un immense merci à Isabelle, qui sait toujours être juste et bienveillante avec tout le monde. J'adresse tous mes remerciements aux doctorants du Cermics. On a vraiment passé de bons moments ensemble, vous avez rendu les journées agréables grâce à des discussions toujours intéressantes et drôles. Un merci particulier à vous avec qui j'ai passé des moments privilégiés qui resteront de super souvenirs : Marion, Alexandre, Benoit, Henri, Adèle, Sébastien, Oumaïma, William, Thomas, Adrien, Maël et tout ceux que j'oublie !

Je souhaite remercier tous mes amis avec qui je passe des moments exceptionnels et qui sont toujours là pour moi. Merci à vous pour ces soirées à refaire le monde autour d'une bière, à rigoler sur des anecdotes qu'on aime se remémorer, à s'écharper dans des débats politiques et footballistiques, ça me fait toujours autant plaisir de vous voir. Merci Thibault, Eliott, Gaspard, Alexis, Vincent, Julien, Robinson, Arnaud, Tom, Adrien, Paul, Théo, Cassandre, Camille, Léa, Jessica, Kathleen, Kim, et tous ceux qui se reconnaîtront.

Merci à ma famille pour tout leur soutien. Ces remerciements mériteraient plus que quelques lignes tant vous comptez pour moi. Merci à mes parents pour l'éducation qu'ils m'ont donné et les valeurs qu'ils m'ont transmis, l'ouverture, l'exigence, la bienveillance et la curiosité. Merci Maman pour ton inaltérable soutien et ton attention à notre bonheur. Merci Papa pour être ce repère stable, juste et humain sur lequel nous pouvons toujours compter. Merci Raphaël pour être notre aîné au grand cœur. Merci Dorian pour nous faire toujours rire et d'amener ton enthousiasme à chaque repas de famille. Merci Joachim pour être toujours présent et un soutien sans faille de Paris à Montréal, en passant par Bangor.

Et enfin merci à Gabrielle d'avoir été si patiente, si compréhensive et de m'avoir apporté tout son soutien chaque jour. C'était long et difficile pour nous deux d'être à distance durant ces trois années, mais comme le dit Albert Camus "C'est cela l'amour, tout donner, tout sacrifier sans espoir de retour". Merci de faire partie de ma vie et de me rendre si heureux.

Abstract

This thesis develops algorithms for stochastic optimization problems such as Markov Decision Processes (MDPs) or Partially Observable Markov Decision Processes (POMDPs), and uses them to give a solution for the airplane maintenance problem at Air France. The research was conducted throughout a scientific chair between Air France and École des Ponts ParisTech.

We introduce a generic predictive maintenance problem for systems with several components evolving over time when a decision maker chooses dynamically which components to maintain at each maintenance slot. His actions are based on partial observations of each component and are linked by capacity constraints. The objective is to find a "memoryless" policy, which is a mapping from observations to actions, minimizing the expected failure costs and maintenance costs over a finite horizon. We formalize such a problem as a weakly coupled POMDP, which models each component as a POMDP. Finding an optimal memoryless policy for the weakly coupled POMDP is difficult for two reasons. First, even when the system has a single component, the problem is already NP-hard. Second, due to the curse of dimensionality, when the number of components grows the POMDP becomes quickly intractable. Our main contributions are mixed-integer linear formulations for POMDPs that give an optimal memoryless policy, as well as valid inequalities that are based on a probabilistic interpretation of the dependences between the random variables. In addition, we introduce an mixed-integer linear formulation that breaks the curse of dimensionality and that induces a "good" policy for weakly coupled POMDP.

In fact, the MDPs and POMDPs lie in the broad class of stochastic optimization problems where the uncertainty is assumed to satisfy a given structure called influence diagram. More precisely, given random variables considered as vertices of an acyclic digraph, a probabilistic graphical model defines a joint probability distribution via the conditional probability distribution of vertices given their parents. In influence diagrams, the random variables are represented by a probabilistic graphical model whose vertices are partitioned into three types: chance, decision and utility vertices. The decision maker chooses the probability distribution of the decision vertices conditionally to their parents in order to maximize his expected utility. Our main contributions are mixed-integer linear formulations for solving the maximum expected utility problem in influence diagrams, as well as valid inequalities, which lead to a computationally efficient algorithm. It generalizes our results on POMDPs to any influence diagrams. We also show that the linear relaxation yields an optimal integer solution for instances that can be solved by the "single policy update," the default algorithm for addressing the maximum expected utility problem in influence diagrams.

The airplane maintenance problem at Air France is a predictive maintenance problem with capacity constraints. Applying the weakly coupled POMDP policy requires to estimate the weakly coupled POMDP parameters. Based on a dataset of historical sensor data, we propose a statistical methodology to cast the airplane maintenance problem as a weakly coupled POMDP. Our approach has the advantage of being interpretable by the maintenance engineers. The numerical experiments show that our maintenance policy give “good” numerical results compared to those obtained by using Air France’s maintenance policy.

Key words: predictive maintenance, partially observable Markov decision processes, probabilistic graphical models, influence diagrams, stochastic optimization, mixed-integer linear programming

Résumé

Cette thèse développe des algorithmes pour des problèmes d'optimisation stochastiques tels que les processus de décision markoviens (MDPs) ou les processus de décision markoviens partiellement observables (POMDPs) [1], et les utilise pour donner une solution au problème de maintenance des avions chez Air France. Cette recherche a été menée dans le cadre d'une chaire scientifique entre Air France et l'École des Ponts ParisTech.

Nous introduisons un problème générique de maintenance prédictive d'un système à plusieurs composants évoluant dans le temps où un décideur choisit dynamiquement les composants à réparer à chaque plage de maintenance. Basées sur des observations partielles de chaque composant, ses actions sont couplées par des contraintes de capacité. L'objectif est de trouver une politique "sans mémoire", c'est-à-dire une application qui associe à l'observation courante (et non l'historique des observations et des actions) une action, qui minimise les coûts de panne et les coûts de maintenance espérés sur un horizon fini. Nous formalisons ce problème sous la forme de POMDPs faiblement couplés où chaque composant est modélisé comme un POMDP.

Trouver une politique optimale sans mémoire pour les POMDPs faiblement couplés est difficile pour deux raisons. Premièrement, même lorsque le système ne comporte qu'un seul composant, le problème est déjà NP-difficile. Deuxièmement, en raison de la malédiction de la dimension, lorsque le nombre de composants augmente, le POMDP devient rapidement insoluble. Nos principales contributions sont des formulations linéaires en nombres entiers pour les POMDPs qui donnent des politiques optimales sans mémoire, ainsi que des inégalités valides qui sont basées sur une interprétation probabiliste des dépendances entre les variables aléatoires du problème. De plus, nous introduisons une formulation linéaire en nombres entiers qui casse la malédiction de la dimension et qui induit une "bonne" politique pour les POMDPs faiblement couplés.

En réalité les MDPs et les POMDPs font partie d'une large classe de problèmes d'optimisation stochastique où l'incertitude satisfait une certaine structure qu'on appelle diagramme d'influence. Plus précisément, en considérant les variables aléatoires comme les sommets d'un digraphe acyclique, un modèle graphique probabiliste définit une distribution de probabilité jointe comme le produit des distributions de probabilités conditionnelles des sommets sachant leurs parents. Dans les diagrammes d'influence, les variables aléatoires sont représentées par un modèle graphique probabiliste dont les sommets sont divisés en trois types : les sommets chances, décisions et utilités. Le décideur choisit la distribution de probabilité des sommets décisions conditionnellement à leurs parents afin de maximiser son utilité espérée. Nos principales con-

[1] Markov Decision Process (MDP) et Partially Observable Markov Decision Process (POMDP) en anglais

tributions sont des formulations linéaires en nombres entiers pour résoudre le problème de l'utilité maximum espérée dans un diagramme d'influence, ainsi que des inégalités valides, qui conduisent à une formulation efficace. Cela généralise nos résultats sur les POMDPs à tout diagrammes d'influence. Nous prouvons également que la relaxation linéaire de notre programme donne une solution optimale en nombres entiers pour les cas qui peuvent être résolus par "single policy update", l'algorithme par défaut pour résoudre le problème de l'utilité maximum espérée dans un diagramme d'influence.

Le problème de maintenance des avions chez Air France est un problème de maintenance prédictive avec des contraintes de capacité. L'application de notre politique pour les POMDPs faiblement couplés produite sur le problème de maintenance des avions à Air France nécessite d'estimer les paramètres des POMDPs faiblement couplés. À partir d'un historique de données brutes enregistrées par des capteurs, nous introduisons une méthodologie statistique qui permet de transformer le problème de maintenance des avions en des POMDPs faiblement couplés. Notre approche a l'avantage d'être interprétable par les ingénieurs de la maintenance. Les expérimentations numériques montrent que notre politique de maintenance donne de "bons" résultats numériques comparés à ceux obtenus en utilisant la politique de maintenance d'Air France.

Mots clés : maintenance prédictive, processus de décision markovien partiellement observable, modèle graphique probabiliste, diagramme d'influence, optimisation stochastique, programmation linéaire en nombres entiers

Contents

List of Figures

List of Tables

1 Introduction

This thesis in Operations Research develops several research topics motivated by a maintenance problem that raised within an industrial partnership with Air France.

Motivation. The quantity of data available on industrial systems has dramatically increased in the last few years. The goal of predictive maintenance is to exploit this data to predict failures, maintain the equipments before they fail, and reduce the overall maintenance costs. The key aspects to model in predictive maintenance strongly depend on the industrial context. For instance, differences in the size of the fleet of components to maintain, the cost of the maintenances, the cost of the failures, or the quality of data available may lead to very different models. In this dissertation, we develop a data-driven optimization framework for the airplane maintenance problem at Air France.

Predictive maintenance for Air France. Airlines performances largely depend on their ability to operate their airplanes as much as possible and reliably. As illustrated on Figure 1.1, airplanes have dense schedules, with several flight legs between each maintenance slot. On recent generations of airplanes, sensors signals are recorded at 1Hz during flights. For each equipment, Air France has access to a collection of time series with different kinds of signals such as pressures, temperatures, intensities, or binary signals. When an airplane arrives in maintenance, Air France uses this data to decide which equipments it should maintain. Maintenance decisions must strike a balance between costs due to over-maintenance and costs due to equipments failures. Besides, since maintenance slots are a rare resource, Air France must prioritize between different equipments.

Research topics discussed in this thesis. The airplane maintenance problem at Air France consists in providing a decision support methodology to derive a *maintenance policy* (see Figure 1.1), which takes in input the sensor data available at the beginning of a maintenance slot, and returns as output which equipments should be maintained. To achieve this goal, we do not have any model or simulator of the equipments behavior, but we have several years of historical data containing signal values and failure dates. The problem we consider is therefore a *data-driven multistage stochastic optimization problem*. It leads us to develop three research lines at the crossroads of Operations Research and Machine learning.

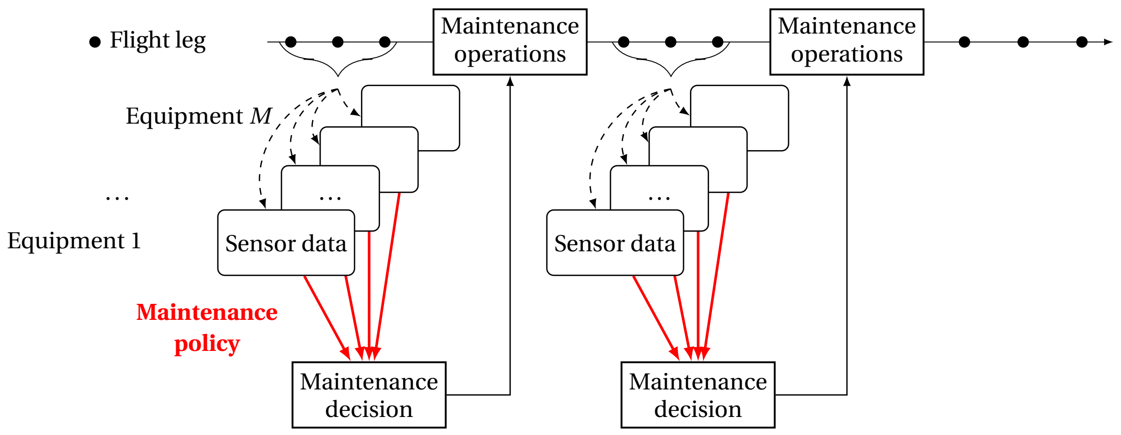


Figure 1.1 – The airplane maintenance problem: we want to find a maintenance policy (red arrows) that takes sensor data from M equipments in input and returns a maintenance decision in output.

A first line focuses on the decision-making process of a generic predictive maintenance problem, which is formalized as a dynamic sequential optimization problem under uncertainty. The decision maker dynamically chooses a restricted number of components to maintain at each maintenance slot. The uncertainty comes from the failures, modeled as random events, which happen during the operating days of the system. Each component is assumed to behave independently but the maintenance decisions on the different components are weakly linked by capacity constraints. This problem lies in the broad class of problems that involve many independent subprocesses that are only weakly linked at each time step. These problems are called *weakly coupled dynamic programs* [2]. A new additional feature from the predictive maintenance problem is the fact that the decisions are based on noisy observations, which makes the system *partially observable*. We propose several methods with guarantees to compute "good" solutions of these optimization problems.

The second line of research that structures this dissertation focuses on the study of stochastic optimization problems where the uncertainty is known to satisfy some structure called *influence diagram*. In particular, this class of problems includes the weakly coupled dynamic program under partial observations induced by the airplane maintenance problem. We propose an exact algorithm, which is based on tools from Operations Research and Machine Learning and which exploits this uncertainty structure.

Finally, the third line of research addresses some *statistical* challenges that come from the practical problem at Air France. Indeed, casting the airplane maintenance problem as a weakly coupled dynamic program with partial observations requires to *learn* a *hidden Markov model* that captures the evolution of the equipment's deterioration over time. In addition, Air France's requirement is that this statistical model has to be interpretable. Based on the dataset of signal values and failure dates on the whole fleet of airplanes, we introduce a methodology that learns such a statistical model that can be interpreted by the maintenance engineers.

Section 1.1 introduces informally the predictive maintenance problem with capacity constraints.

Section 1.2 summarizes our main mathematical contributions to the optimization problem of weakly coupled dynamic programs with partial observations. Section 1.3 details our theoretical contributions to the optimization problem in influence diagrams. Section 1.4 describes our contributions that address the statistical challenges raised by the airplane maintenance problem.

While this dissertation starts in Part I by investigating solution algorithms for these weakly coupled dynamic programs with partial observations, Part III details how we exploit those in the airplane maintenance problem at Air France. Part II contains our work on the optimization problem in influence diagrams. Each chapter of this dissertation contains a section with bibliographical remarks.

1.1 The predictive maintenance problem with capacity constraints

In the industry, planning the maintenance of a system consists in choosing when to intervene during the system's operating period and which actions should be carried out during this intervention. In our case, dates of the maintenance are already scheduled by a flight scheduling tool. Thus, the predictive maintenance problem we consider focuses on the second issue of the planning problem, i.e., deciding which actions to take at each scheduled maintenance slot.

The system we consider has multiple components, each of them evolving over time. In general, the systems considered are mechanical, which ensures that each component deteriorates over time and fails after several operating hours/days. The failures happen during the operating days of the components of the system between two maintenance slots. At each maintenance slot, the decision maker chooses which components to maintain and his choices are based on the observed *degradation state* of each component. This degradation state characterizes the different performance rates of the component and is described through a discrete indicator, whose values range from the state "perfect functioning" to the state "failure." The deterioration process of a component is *stochastic* in the sense that the component transits from a degradation state s to a more critical degradation state s' according to a *probability distribution*. An additional widely used assumption in the industry is to consider that this deterioration is *Markovian*, i.e., the state at a time only depends on the state at the previous time. Figure 1.2 illustrates the modeling of the deterioration process of a component.

However, in many practical problems the decision maker has only access to a *partial observation* of the degradation state of a component, which is also a discrete noisy indicator. Indeed, this indicator usually corresponds to error messages resulting from the detection of an abnormal behavior of physical measurements of the components. It makes this partial observation very sensitive to measurement errors and noisy regarding to the degradation state. Figure 1.2 illustrates these noisy observations by the squiggly blue arrows. Such a stochastic model is called a *Hidden Markov Chain*. It follows that at each maintenance slot the decision is taken based on the partial observations of the components instead of the degradation states as shown in Figure 1.3.

Between two maintenance slots, the components are assumed to evolve *independently*, i.e., the deteriorating process of a component does not influence the deterioration process of the others. In an industrial context, the components are linked by the maintenance decisions through

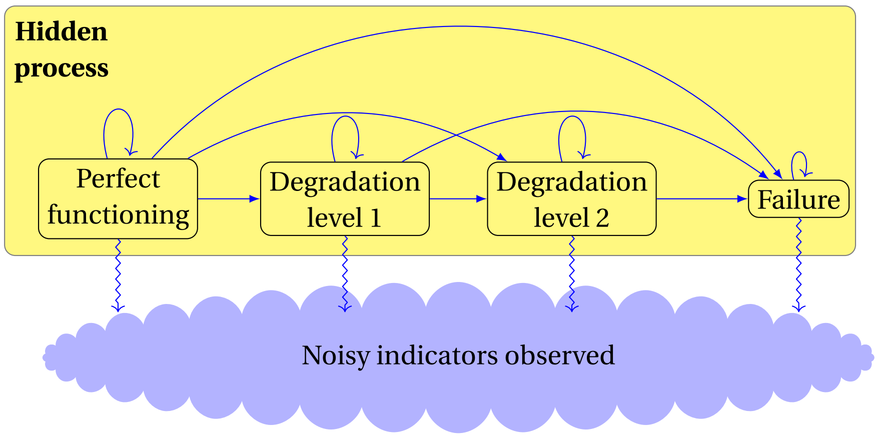


Figure 1.2 – Degradation state evolution of a component with four states. The component starts in its most healthy state and evolves stochastically (blue arrows) toward a more degraded state. At each degradation state, the decision maker has access to a noisy observation that is randomly emitted (squiggly blue arrows).

capacity constraints. This is the case of our applied problem: To be feasible, the maintenance decisions of the airplane maintenance problem have to satisfy the following constraint.

The number of maintained components at each maintenance slot is not greater than K.

While considering the maintenance problem of each component independently is easier, these constraints force the decision maker to consider the evolution of the whole system. If a component is maintained, then its degradation state recovers to "perfect functioning." Otherwise, it keeps on evolving to a more degraded state. Maintaining a component leads to a *maintenance cost* that is significantly lower than the *failure cost*.

Figure 1.3 summarizes the predictive maintenance problem with capacity constraints. Choosing a maintenance policy consists in choosing at each future maintenance slots, for each possible partial observation, a maintenance decision satisfying the capacity constraints. Then, the goal of the decision maker is to choose a maintenance policy that minimizes the expected costs expressed as the sum of the maintenance costs and the failure cost. Chapter 3 is devoted to formalize mathematically the predictive maintenance problem with capacity constraints. It raises two scientific challenges: How do we cast the airplane maintenance problem as a predictive maintenance with capacity constraints (Section 1.4), and, how do we find a "good" maintenance policy (Section 1.2)?

1.2 Decision processes with partial observations

We start our discussion on choosing a "good" maintenance policy by introducing an adequate mathematical model for the predictive maintenance problem with capacity constraints. Since the deterioration process of the component is Markovian and the decision maker has only access to a partial observation, it is natural to formalize the problem within the *Partially Observ-*

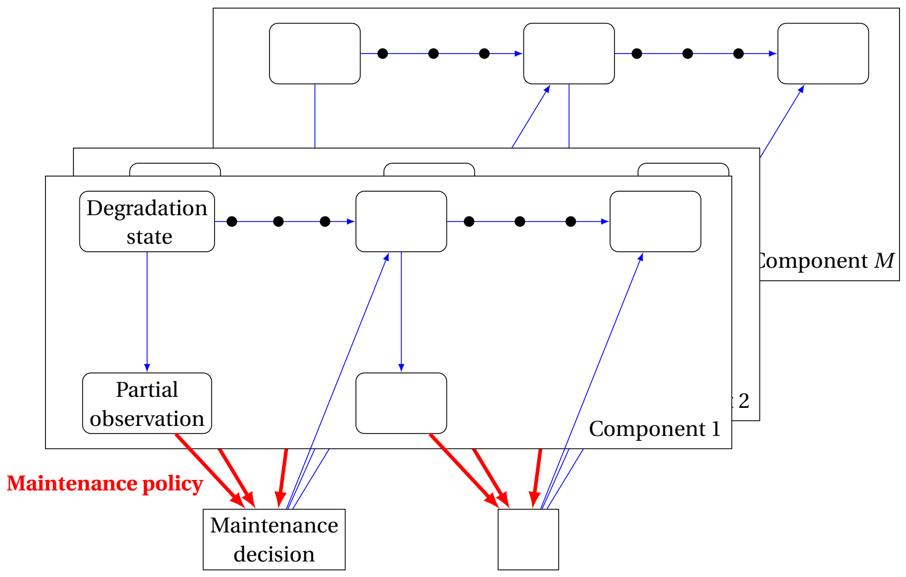


Figure 1.3 – Modeling of the predictive maintenance problem with capacity constraints for a system with M components. The blue and red arrows respectively represent the stochastic deterioration of the components and the maintenance policy. At each maintenance slot, each component is in a degradation state and the decision maker has access to a partial observation on each component. Then, the decision maker applies a maintenance policy (red arrows) that leads to a maintenance decision that impacts the degradation state of each component.

able Markov Decision Process (POMDP) framework.

The POMDP framework. We start by considering a system with one component. In such problems, at each time step, the component is in a *state* s in some finite state space $\mathcal{X}_S$. The decision maker does not observe s, but has access to an *observation* o that belongs to some finite observation space $\mathcal{X}_O$, and is randomly emitted with probability $p(o|s)$. Based on this observation, the decision maker chooses an action a from some finite action space $\mathcal{X}_A$. The component then transits randomly to a new state s' in $\mathcal{X}_S$ with probability $p(s'|s,a)$ and the decision maker obtains an immediate reward $r(s,a,s')$. The goal of the decision maker is to find a policy $\delta^t_{a|o}$, which represents a conditional probability of taking action a in $\mathcal{X}_A$ given the observation o at time t, maximizing the expected total reward over a finite horizon T.

The problem on which we focus is

$$\max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right], \tag{1.1}$$

where S_t, O_t and A_t are random variables representing the state, the observation and the action at time t. The expectation in (1.1) over $\boldsymbol{\delta}$ is taken according to the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$

induced by the policy $\boldsymbol{\delta}$ chosen in Δ, and defined as follows

$$\mathbb{P}_{\boldsymbol{\delta}}\big((S_t = s_t, O_t = o_t, A_t = a_t)_{1\leqslant t\leqslant T}, S_{T+1} = s_{T+1}\big) = p(s_1)\prod_{t=1}^{T} p(o_t|s_t)p(s_{t+1}|s_t,a_t)\delta^t_{a_t|o_t} \tag{1.2}$$

for every $\big((s_t,o_t,a_t)_{1\leqslant t\leqslant T}, s_{T+1}\big)$ in $\big(\mathcal{X}_S \times \mathcal{X}_O \times \mathcal{X}_A\big)^T \times \mathcal{X}_S$.

Problem (1.1) is known as the POMDP problem with *memoryless* policies, meaning that the action taken at a time is only based on the current observation instead of the history of observations and actions. POMDPs generalize the *Markov Decision Processes* (MDPs), where the decision maker has access to the state of the system. While MDPs can be solved in polynomial time, the POMDP problem with memoryless policies is NP-hard. Unless P=NP, the fact that the decision maker has only access to a partial observation makes the problem harder to solve.

One contribution in this dissertation is an *Mixed-Integer Linear Program* (MILP), which is a standard tool in Operations Research, that provides an optimal policy of the POMDP problem with memoryless policies. It formulates an optimization problem described by a linear objective function, linear constraints and integer variables with the following canonical form:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathbf{c}^{\mathbf{T}}\mathbf{x} \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} \leqslant \mathbf{b} \\
& \mathbf{x} \geqslant 0 \\
& \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p},
\end{aligned}$$

where **b** and **c** are real vectors and **A** is a real matrix. While the MDP maximization problem can be solved using a *linear program* (see, e.g., [124]), i.e., by relaxing the integrality constraints, we use integer variables to address the POMDP maximization problem (1.1). Such integer formulations can be solved efficiently using off-the-shell optimization solvers. Since these solvers use the *Branch-and-Bound* algorithm, it is natural to derive *valid inequalities* that improve the quality of the bounds obtained when solving the problem without integrality constraints. A second contribution is a collection of valid inequalities that help the resolution of the MILP formulation. Numerical experiments show its efficiency.

The POMDP framework for the predictive maintenance problem with capacity constraints. It is natural to assume that each component behaves as a POMDP. For each component the degradation state, the partial observation and the maintenance decision (see Figure 1.3) are respectively represented by the state, the observation and the action of a POMDP. For each component, all the conditional probabilities $p(o|s)$ and $p(s'|s,a)$ are computed using the parameters of the hidden Markov model. Given a system with M components, the problem consists in M subproblems that are *weakly* linked by the capacity constraints that affect the actions taken on each component. For instance, the capacity constraint "*the number of maintained components at each maintenance slot is not greater than K*" is modeled using the constraint $\sum_{m=1}^{M} a^m \leqslant K$, where a^m is the action on component m and equal to 1 if component m is maintained and 0 otherwise. Since the actions taken on each component are linked, a maintenance policy has to be considered on the whole system. At each time t, the maintenance policy is of the form $\delta^t_{\mathbf{a}|\mathbf{o}}$, where $\mathbf{a} = (a^1, \dots, a^M)$ and $\mathbf{o} = (o^1, \dots, o^M)$ respectively represent the action and observation of the system. For each component m, a transition (s, a, s') leads to an immediate reward defined as follows

$$r^m(s,a,s') = -\mathrm{Cost}^m(\mathrm{maintenance})\mathbb{1}_{\mathrm{maintain}}(a) - \mathrm{Cost}^m(\mathrm{failure})\mathbb{1}_{\mathrm{failure}}(s'),$$

where $\mathbb{1}_x(y)$ is equal to 1 if $x = y$ and 0 otherwise.

If the decision maker has access to the degradation state of each component, then each subproblem reduces to a Markov decision process and the problem on the whole system is a *weakly coupled MDP* [105]. In our case, the fact that the decision maker has only access to a partial observations requires to extend this notion to *weakly coupled POMDPs*. In addition to the partially observable aspect, a second difficulty when solving such a problem comes from the *curse of dimensionality*. Indeed, the size of the system space grows exponentially with the number of components of the system, which makes the usual algorithms computationally expensive. To figure out this difficulty, one observes that encoding a maintenance policy $\boldsymbol{\delta}$ requires a table of the size roughly equals to $T \prod_{m=1}^M |\mathcal{X}_O^m||\mathcal{X}_A^m|$, which is in general intractable.

To get around this difficulty, we introduce the notion of *implicit* policy. In opposition to *explicit* policies that fully encode the policy in the solution of a single optimization problem, the implicit policies are the ones for which given an observation $\mathbf{o}$, each vector $(\delta^t_{\mathbf{a}|\mathbf{o}})_{\mathbf{a}}$ is computed through a tailored optimization problem. It has a practical interest because the decision maker does not need to compute the policies for all possible observations. Indeed, once an observation $\mathbf{o}$ is revealed at time t, it only requires to compute the vector $(\delta^t_{\mathbf{a}|\mathbf{o}})_{\mathbf{a}}$. Another contribution described in Part I of this dissertation is an mixed-integer linear formulation that leverages the one we introduce for the POMDP problem with memoryless policies and that is used to compute a “good” implicit policy for the weakly coupled POMDP problem. Like the formulations of Adelman and Mersereau [2] for weakly coupled MDPs, the main advantage of our formulation is to contain a polynomial number of variables and constraints, which breaks the curse of dimensionality. Again, we improve the formulation by adding a collection of inequalities in our MILP formulation. In addition, we provide theoretical guarantees about the optimal value of our MILP formulation by computing a *lower bound* and an *upper bound*.

1.3 Decision making with structured uncertainty

MDPs and POMDPs (Section 1.2) are specific cases of discrete stochastic optimization problems where the probability distribution of the random variables is assumed to satisfy some structure specified through an *influence diagram*. Influence diagrams form a flexible tool that enables to model a large class of stochastic optimization problems where the “nature” is assumed to be structured. We introduce the optimization problem in influence diagrams through a toy example.

The example is illustrated in Figure 1.4a. Consider a patient who consults a doctor. The patient suffers from a disease D, which the doctor wants to diagnose. This disease appears on a patient with probability $\mathbb{P}(D)$, and leads to observable clinical symptoms represented by S, which are randomly emitted with probability $\mathbb{P}(S|D)$. Based on the observation of the symptoms S, the doctor has to decide which treatment T to apply on the patient. The treatment T and the disease D give a response R_0 of the patient with probability $\mathbb{P}(R_0|T,D)$, which leads to a reward $r_0(R_0)$ indicating the presence of undesirable side effects. Then, based on the response R_0, the doctor decides if the disease requires a stronger medical intervention I. It finally gives a result R_1 with probability $\mathbb{P}(R_1|I,D)$, indicating if the intervention on the disease D was a success, and it leads to a reward $r_1(R_1)$. The disease D, the symptoms S, the responses R_0 and R_1, are represented by random variables that are not controlled by the doctor. They are called *chance variables* because their values are chosen by the "nature," i.e., everything that is not controlled by the doctor. In opposition to chance variables, the treatment T and the medical intervention I are represented by random variables called *decision variables*, whose values get to be chosen by the doctor. The functions r_0 and r_1 are the *utility functions* that depend respectively on the responses R_0 and R_1. The goal of the doctor is to choose a strategy $\boldsymbol{\delta} = \big(\delta_T(T|S), \delta_I(I|R_0)\big)$ maximizing his total expected utility $\mathbb{E}_{\boldsymbol{\delta}}\big[r_0(R_0) + r_1(R_1)\big]$ where the expectation is taken according to the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}(D,S,T,R_0,I,R_1) = \mathbb{P}(D)\mathbb{P}(S|D)\delta_T(T|S)\mathbb{P}(R_0|T,D)\delta_I(I|R_0)\mathbb{P}(R_1|I,D)$.

This type of problem can be encoded graphically using a *directed acyclic graph*, whose set of vertices V contains three types of vertices: chance (V^{c}), decision (V^{a}) and utility vertices (V^{r}) corresponding respectively to chance variables, decision variables and utility functions. An influence diagram, introduced by Howard and Matheson [56], is a directed acyclic graph over these vertices such that there are no outgoing arcs from a utility vertex. Figure 1.4a illustrates the influence diagram that models the medical decision problem we described. Each chance variable X_v, represented by a chance vertex v, is associated with a conditional probability distribution $\mathbb{P}(X_v|X_{\mathrm{pa}(v)})$ of X_v given $X_{\mathrm{pa}(v)}$, where $X_{\mathrm{pa}(v)}$ is the vector of random variables represented by the parents of v in the influence diagram, the tails of the incoming arcs. Each utility function r_v, represented by a utility vertex v, is associated with a deterministic function r_v that maps each instantiation $X_{\mathrm{pa}(v)}$ to a real value. The choices of the decision maker are then modeled using a *strategy* $\boldsymbol{\delta} = (\delta_v)_{v\in V^{\mathrm{a}}}$ that is a vector of conditional probability distributions $\delta_v(X_v|X_{\mathrm{pa}(v)}) = \mathbb{P}(X_v|X_{\mathrm{pa}(v)})$. Given a strategy $\boldsymbol{\delta}$, the joint probability distribution over all the random variables writes down

$$\mathbb{P}_{\boldsymbol{\delta}}(X_{V^{\mathrm{c}}\cup V^{\mathrm{a}}} = x_{V^{\mathrm{c}}\cup V^{\mathrm{a}}}) = \prod_{v\in V^{\mathrm{c}}} \mathbb{P}(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}) \prod_{v\in V^{\mathrm{a}}} \delta_v(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}). \tag{1.4}$$

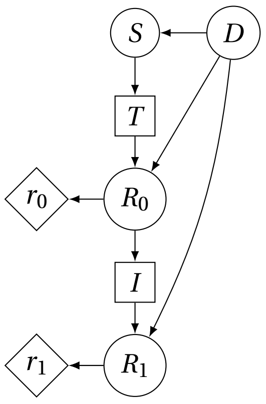
The problem solved by the decision maker is the *Maximum Expected Utility* problem

$$\max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{v\in V^{\mathrm{r}}} r_v(X_{\mathrm{pa}(v)})\right], \tag{1.5}$$

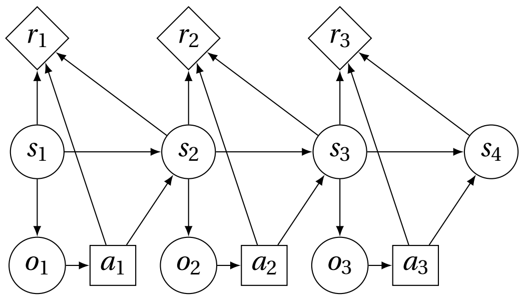
where Δ is the set of possible conditional probability distributions for each decision vertex.

The modeling power of an influence diagram can also be observed on the POMDPs described in Section 1.2. The chance variables are the states $(S_t)_{1\leqslant t\leqslant T+1}$ and observations $(O_t)_{1\leqslant t\leqslant T}$, the decision variables are the actions $(A_t)_{1\leqslant t\leqslant T}$, and finally the utility variables are the rewards $\big(r(S_t,A_t,S_{t+1})\big)_{1\leqslant t\leqslant T}$. The probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ of (1.2) can be written as (1.4) using the influence diagram represented in Figure 1.4b. The goal is to choose $\boldsymbol{\delta}$ maximizing the expected total reward $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{t=1}^T r(S_t,A_t,S_{t+1})\big]$ where the expectation is taken according to the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$. This problem can be represented using the influence diagram in Figure 1.4b.

It leads us to the question of finding a strategy that solves the maximum expected utility prob-



(a) The medical decision making problem



(b) A POMDP with memoryless policy

Figure 1.4 – The two influence diagrams modeling the medical example and the POMDP example. The circle, square and diamond vertices respectively represent the chance, decision and utility vertices.

lem in an influence diagram. This question raises two scientific challenges. First, evaluating a given strategy is already difficult. The difficulty of evaluating a strategy is the difficulty of solving the *inference problem* in an influence diagram, whose difficulty is exponential in the *treewidth*. Given a feasible strategy $\boldsymbol{\delta}$, a subset of vertices $C \subseteq V$, the inference problem consists in computing the marginal probability $\mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C)$. Second, optimizing over the set of strategies Δ is also difficult because of the huge number of possible vectors of conditional probability distributions.

Part II focuses on addressing this second challenge. It generalizes the results obtained for the POMDPs of Section 1.2 to any maximum expected utility problem in an influence diagram. We emphasize that the results we obtain are based on the combination of several tools from *probabilistic graphical model theory* and *integer programming*. A first contribution described in Part II is an mixed-integer linear program that gives an optimal strategy of the maximum expected utility problem in influence diagrams. While the usual solutions are based on *dynamic programming* like algorithms along the graph (see, e.g., [76, 81, 89]), our approach is based on mathematical programming. A second contribution is a collection of valid inequalities that improves our integer linear formulation. A third contribution is a characterization of the "easy" case of influence diagrams, i.e., the influence diagrams such that the maximum expected utility problem becomes tractable. The results in Part II have been partially published in Parmentier et al. [117] and some of their proofs are based on other technical results in the preprint Cohen and Parmentier [27]. Several results have been added after the publication of this paper.

1.4 Statistical methodology for the airplane maintenance problem

Finally we wish to use of the policy of weakly coupled POMDP mentioned in Section 1.2 in our applied problem. However, modeling the maintenance problem of a real-life system as a predictive maintenance problem as in Section 1.1 is not immediate. It is particularly the case of the airplane maintenance problem at Air France. Indeed, the available data at each maintenance slots are the sensors data, which correspond to the signal values recorded during flights. An example of sensor signal is the value of the pressure in an equipment recorded at 1Hz during a flight. Hence, our dataset contains several years of sensor data and several failure dates. Consequently, expressing the airplane maintenance problem as a predictive maintenance problem with capacity constraints requires to compute the hidden Markov models of the equipment deterioration processes from our dataset. We propose a preliminary statistical work to cast the airplane maintenance problem (Figure 1.1) as a predictive maintenance problem with capacity constraints (Figure 1.3).

It raises three practical issues. First, in the predictive maintenance problem with capacity constraints the decision maker has access to a discrete indicator as partial observation, while the observations in the airplane maintenance problem correspond to sensor data, which are continuous and high-dimensional. Hence, it requires to develop a methodology that enables to transform the signal values into discrete indicators.

The second issue concerns the *learning* phase. Based on the historical data, the aim is to estimate the parameters that fully define the adequate hidden Markov model for each equipment, which is required as inputs of the predictive maintenance problem with capacity constraints.

The third issue is about the *interpretability* of the approach. Indeed, even if they do not impact the safety of the flights,[1] the maintenance decisions we support are not benign. Maintenance operations such as unmounting the landing gear of a long-haul airplane are expensive and time consuming. And if a failure happens, Air France will have to cancel the next flights operated by the airplane, which is even more expensive. Thus, if we want our model to be used in practice, the predictions of the statistical model, the hidden Markov model, as well as the maintenance decisions resulting from a maintenance policy, must be trusted by the maintenance engineers. An additional difficulty in this point is the fact that the dataset we consider contains a small amount of failures because Air France tries to avoid them as much as possible and we have a small amount of sensor data of the behavior of an equipment right before it fails. Consequently, we cannot validate the statical model using experimental validations.

Our approach is described in Chapter 10 and addresses these issues by combining four steps that use several statistical tools. First, since the data we handle are high-dimensional and noisy, we transform the collection of time series recorded during a flight by a reasonable number of relevant *features*. For instance, if a sensor records the evolution of the pressure in an equipment during a flight, an example of feature is the average or the standard deviation of the pressure over the flight length. Second, we learn the parameters of a *Gaussian Hidden Markov model* that predicts the evolution of the vector of features. Such a model is a hidden Markov model where the observations are continuous and are emitted according to a Gaussian probability distribution. Third, we transform this Gaussian hidden Markov model into a hidden Markov model with discrete observations using a *decision tree*. It takes a vector of continuous observations in input and returns a discrete indicator. The mechanism inside the decision tree is a combination of binary splitting rules, such as “is the average pressure above 20 bars?,” that leads to a discrete indicator, whose values correspond to different failure risk levels.

These binary rules can be easily understood and validated by maintenance engineers, which

[1] A critical equipment fails when it reaches a conservative threshold set by regulation agencies, and beyond which the airplanes are not allowed to take off again.

addresses the third issue. Since this indicator is discrete, it also addresses the first issue. Finally, based on the Gaussian hidden Markov model and the decision tree we compute the parameters of the hidden Markov model with the discrete indicators, which addresses the second issue. Using this approach enables to cast the airplane maintenance problem at Air France as a predictive maintenance problem with capacity constraints. The numerical experiments show that our maintenance policy give “good” numerical results compared to those obtained by using Air France’s maintenance policy.

2 Introduction (Français)

Plusieurs thématiques de Recherche Opérationnelle sont développées dans cette thèse et motivées par un problème de maintenance de notre partenaire industriel Air France.

Contexte. La quantité de données disponibles sur les systèmes industriels a énormément augmenté durant les dernières années. Le but de la maintenance prédictive est d'exploiter ces données afin de prédire les pannes, de réparer les équipements avant qu'ils ne tombent en panne et de réduire le coûts globaux de maintenance. Les aspects clés de la modélisation d'un problème de maintenance prédictive dépendent essentiellement du contexte industriel. Par exemple, les différences dans le nombre d'équipements à entretenir, les coûts de maintenance, les coûts de panne, ou la qualité des données disponibles peuvent amener des modélisations très différentes. Dans cette thèse, nous développons un cadre d'optimisation guidée par les données pour le problème de maintenance des avions chez Air France.

La maintenance prédictive pour Air France. Les performances des compagnies aériennes dépendent de la capacité à assurer tous les vols avec leurs avions de manière fiable. Comme on peut l'observer sur la figure 1.1, les planning horaires des avions sont denses, avec plusieurs vols entre chaque plage de maintenance. Sur les avions de nouvelle génération, des signaux échantillonnés à 1Hz sont enregistrés par des capteurs durant les vols. Pour chaque équipement, Air France a accès à un ensemble de séries temporelles qui correspondent à différents types de signaux comme des températures, des pressions, des intensités, ou des signaux binaires. Lorsqu'un avion arrive en maintenance, Air France utilise ces données afin de choisir quels équipements doivent être réparés. Les décisions de maintenance doivent assurer un équilibre entre les coûts de sur-maintenance et les coûts dus aux pannes sur les équipements. En outre, comme les plages de maintenance sont rares, Air France doit prioriser entre les équipements.

Les sujets de recherche traités dans cette thèse. Le problème de maintenance des avions chez Air France est de proposer une méthodologie d'aide à la décision pour trouver une *politique de maintenance* (en rouge sur la figure 2.1), qui prend en entrée les données brutes disponibles au début de la plage de maintenance, et retourne en sortie les équipements à réparer. Pour cela, nous n'avons pas de modèle ou de simulateur de l'évolution des équipements, mais nous avons accès à un historique de plusieurs années de données brutes et quelques dates de panne.

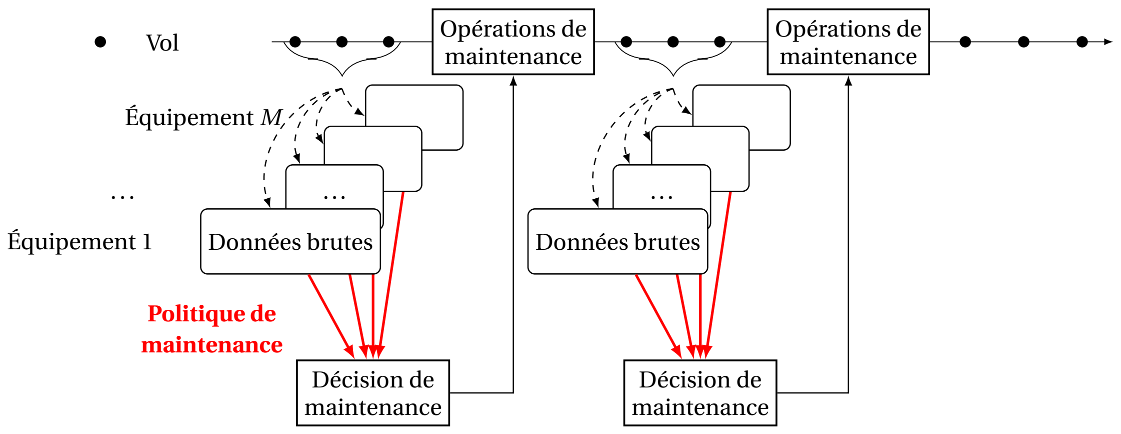


Figure 2.1 – Le problème de maintenance d'un avion : on cherche une politique de maintenance (flèches rouges) qui prend en entrée les données brutes de M équipements et retourne la décision de maintenance en sortie.

Le problème que nous considérons est un *problème d'optimisation stochastique multi-étapes guidée par les données*. Cela nous mène vers trois axes de recherche au carrefour de la recherche opérationnelle et l'apprentissage machine.

Un premier axe de recherche s'intéresse au processus de décision d'un problème de maintenance prédictive que nous formalisons comme un problème d'optimisation séquentiel avec incertitudes. Pour chaque plage de maintenance, le décideur choisit dynamiquement un nombre restreint de composants à réparer. L'incertitude vient des pannes, qu'on modélise comme des événements aléatoires qui peuvent apparaître durant les jours de fonctionnement du système. On suppose que les composants évoluent de manière indépendante mais les décisions de maintenance prises sur chacun des composants sont liées par des contraintes de capacité. Ce problème appartient à une large classe de problèmes d'optimisation séquentielle avec plusieurs processus indépendants qui sont faiblement liés à chaque pas de temps par les décisions. Ces problèmes sont appelés *programmes dynamiques faiblement couplés* [2]. Une particularité du problème de maintenance prédictive est que les décisions sont prises à partir d'observations bruitées, ce qui rend le système *partiellement observable*. Nous proposons plusieurs méthodes avec des garanties qui calculent de “bonnes” solutions pour ces problèmes d'optimisation.

Le second axe de recherche qui structure cette thèse se porte sur l'étude des problèmes d'optimisation stochastique où l'aléa a une certaine structure appelée *diagramme d'influence*. Cette classe de problème contient les programmes dynamiques faiblement couplés avec observations partielles qui modélisent le problème de maintenance des avions. Nous proposons un algorithme exact basé sur des outils de recherche opérationnelle et d'apprentissage machine exploitant la structure de l'aléa.

Enfin, le troisième axe de recherche aborde des questions *statistiques* qui viennent du problème pratique d'Air France. En effet, transformer le problème de maintenance des avions comme des programmes dynamiques faiblement couplés avec observations partielles nécessite d'*apprendre* un modèle statistique qui représente l'évolution de la détérioration de chaque équipement au cours du temps. De plus, les exigences d'Air France imposent que ce modèle statistique soit interprétable. À l'aide d'une base de données brutes et de plusieurs dates de panne sur l'ensemble de la flotte d'avions, on propose une méthodologie qui apprend un modèle statistique qui peut être interprété par les ingénieurs de la maintenance.

La section 2.1 introduit de manière informelle le problème de maintenance prédictive avec contraintes de capacité. La section 2.2 résume nos principales contributions pour le problème d'optimisation des programmes dynamiques faiblement couplés avec observations partielles. La section 2.3 détaille nos contributions théoriques au problème d'optimisation dans les diagrammes d'influence. La section 2.4 décrit nos contributions pour résoudre les questions statistiques qui proviennent du problème de maintenance des avions.

Cette thèse commence en partie I par proposer des algorithmes pour les programmes dynamiques faiblement couplés avec observations partielles, alors que la partie III explique comment nous les exploitons pour le problème de maintenance des avions chez Air France. Le contenu de la partie II correspond à nos travaux sur le problème d'optimisation dans les diagrammes d'influence. A la fin de chaque chapitre de cette thèse, on trouvera une section contenant des remarques bibliographiques en rapport avec le sujet du même chapitre.

2.1 Le problème de maintenance prédictive avec contraintes de capacité

Dans l'industrie, planifier la maintenance d'un système revient à choisir quand on souhaite intervenir sur le système durant ses périodes de fonctionnement et choisir quelles actions réaliser durant ces interventions. Dans notre cas, les dates de maintenance de chaque avion sont déjà fixées par le programme de vol. C'est pourquoi le problème de maintenance prédictive que nous considérons se concentre seulement sur la deuxième problématique : choisir les actions à prendre durant des plages de maintenance déjà fixées.

Le système que nous considérons contient plusieurs composants qui évoluent chacun au cours du temps. En général, les systèmes considérés sont mécaniques. Cela nous assure que chaque composant se détériore au cours du temps et tombe en panne après plusieurs heures/jours de fonctionnement. Les pannes apparaissent durant les jours de fonctionnement des composants du système entre deux plages de maintenance. À chaque plage de maintenance, le décideur choisit les composants à réparer et ses choix sont basés sur une observation de l'*état de dégradation* de chaque composant. Cet état de dégradation caractérise les différents modes de performance de chaque composant et est représenté par un indicateur discret dont les valeurs vont de l'état "fonctionnement optimal" à l'état "panne". Le processus de détérioration d'un composant est *stochastique* dans le sens où un composant passe d'un état de dégradation s à un état de dégradation plus critique s' selon une *distribution de probabilité*. Une hypothèse souvent utilisée dans l'industrie est de supposer que le processus de détérioration est *markovien*, c-à-d, l'état actuel ne dépend que l'état au temps précédent. La figure 2.2 illustre la modélisation du processus de détérioration d'un composant.

Cependant, dans beaucoup de problèmes pratiques le décideur a uniquement accès à une *observation partielle* de l'état de dégradation d'un composant, qui est représenté par un indica-

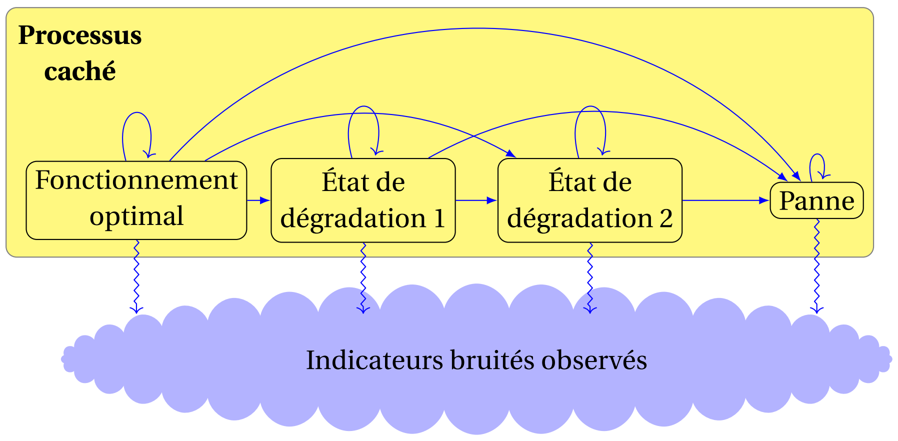


Figure 2.2 – L'évolution de l'état de dégradation d'un composant avec quatre états. Le composant débute dans son état le plus sain et évolue de manière stochastique (flèches bleues) vers un état plus dégradé. Pour chaque état de dégradation, le décideur a accès à une observation bruitée qui est émise de manière aléatoire (flèches bleues en zigzag).

teur discret bruité. En effet, cet indicateur correspond souvent à des messages d'erreurs qui résultent de la détection d'un comportement anormal des mesures physiques sur les composants. Cela rend l'observation partielle très sensible aux erreurs de mesure, et bruitée par rapport à l'état de dégradation. Les flèches bleues en zigzag représentent ces observations bruitées sur la figure 2.2. Ce modèle stochastique est une *chaîne de Markov cachée*. C'est pour cela qu'à chaque plage de maintenance, la décision est basée sur les observations partielles des composants au lieu des états de dégradation comme sur la figure 2.3.

Les composants évoluent de manière *indépendante* entre deux plages de maintenance, c-à-d, le processus de détérioration d'un composant n'influence pas le processus de détérioration des autres composants. Dans un contexte industriel, les composants sont couplés par les décisions de maintenance qui doivent respecter des contraintes de capacité. C'est le cas de notre problème : les décisions de maintenance du problème de maintenance des avions doivent satisfaire la contrainte suivante

Le nombre de composants qu'on peut réparer durant une plage de maintenance ne peut dépasser K.

Alors qu'il serait plus simple de considérer le problème de maintenance de chacun des composants indépendamment, ces contraintes forcent le décideur à considérer l'évolution du système complet. Lorsqu'on répare un composant, on suppose que son état de dégradation retourne à l'état de “fonctionnement optimal”. Sinon, il continue à évoluer vers un état plus dégradé. Réparer un composant amène un *coût de maintenance* qui est significativement moins élevé que le *coût de panne*.

Le problème de maintenance prédictive avec contraintes de capacité est résumé sur la figure 2.3. Choisir une politique de maintenance revient à choisir pour chaque plage de main-

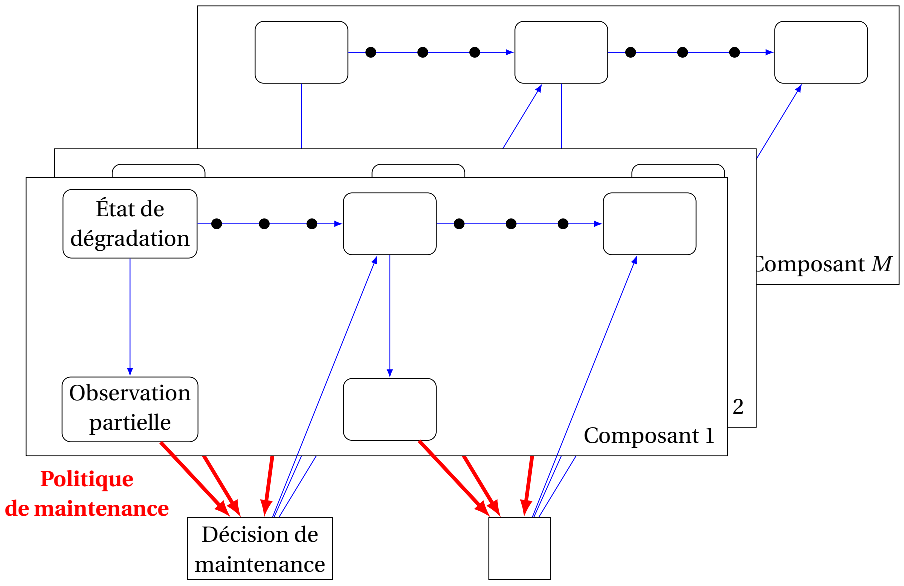


Figure 2.3 – Modélisation du problème de maintenance prédictive avec contraintes de capacité pour un système de M composants. Les flèches bleues et rouges représentent respectivement la détérioration stochastique des composants et la politique de maintenance. À chaque plage de maintenance, chaque composant est dans un état de dégradation et le décideur a accès à une observation partielle pour chaque composant. Puis le décideur applique une politique de maintenance (flèches rouges) qui retourne une décision de maintenance qui impacte l'état de dégradation de chaque composant.

tenance, pour chaque observation partielle possible, une décision de maintenance qui satisfait les contraintes de capacité. Alors, l'objectif du décideur est de choisir une politique de maintenance qui minimise l'espérance des coûts qui s'écrivent comme la somme des coûts de maintenance et des coûts de panne. Dans le chapitre 3 on formalise mathématiquement le problème de maintenance prédictive avec contraintes de capacité. Cela soulève alors deux questions scientifiques : Comment transforme-t-on le problème de maintenance des avions en un problème de maintenance prédictive avec contraintes de capacité (section 2.4), et, comment trouve-t-on une "bonne" politique de maintenance (section 2.2) ?

2.2 Processus de décisions avec observations partielles

Nous débutons notre discussion sur le choix d'une "bonne" politique de maintenance en introduisant une modélisation mathématique adéquate pour le problème de maintenance prédictive avec contraintes de capacité. Comme le processus de détérioration d'un composant est markovien et que le décideur a seulement accès à une observation partielle, on formalise de manière naturelle le problème comme un *Processus de Décision Markovien Partiellement Observable* (POMDP)[1].

[1] POMDP pour Partially Observable Markov Decision Process en anglais.

Le cadre des POMDPs. On commence par considérer un système avec un composant. Pour de tels problèmes, à chaque pas de temps, le composant est dans un *état* s qui appartient à un espace d'états fini $\mathcal{X}_S$. Le décideur n'observe pas s, mais a accès à une *observation* o qui appartient à un espace d'observation fini $\mathcal{X}_O$, et qui est émis aléatoirement avec probabilité $p(o|s)$. À partir de cette observation, le décideur choisit une action a qui appartient à un espace d'action fini $\mathcal{X}_A$. Le composant évolue aléatoirement vers un nouvel état s' dans $\mathcal{X}_S$ avec probabilité $p(s'|s,a)$ et le décideur reçoit une récompense $r(s,a,s')$. L'objectif du décideur est de trouver une politique $\delta^t_{a|o}$, qui représente la probabilité conditionnelle de prendre une action a dans $\mathcal{X}_A$ sachant l'observation o au temps t, qui maximise l'espérance totale de la récompense jusqu'à un horizon fini T.

Le problème auquel nous nous intéressons est

$$\max_{\boldsymbol{\delta}\in\Delta}\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t,A_t,S_{t+1})\right], \tag{2.1}$$

où S_t, O_t et A_t sont des variables aléatoires qui représentent l'état, l'observation et l'action au temps t. L'espérance dans l'expression (2.1) est prise selon la distribution de probabilité $\mathbb{P}_{\boldsymbol{\delta}}$ induite par une politique $\boldsymbol{\delta}$ choisie dans Δ, et définie de la manière suivante

$$\mathbb{P}_{\boldsymbol{\delta}}\big((S_t=s_t,O_t=o_t,A_t=a_t)_{1\leqslant t\leqslant T},S_{T+1}=s_{T+1}\big)=p(s_1)\prod_{t=1}^{T}p(o_t|s_t)p(s_{t+1}|s_t,a_t)\delta^t_{a_t|o_t} \tag{2.2}$$

pour tout $\big((s_t,o_t,a_t)_{1\leqslant t\leqslant T},s_{T+1}\big)$ dans $\big(\mathcal{X}_S\times\mathcal{X}_O\times\mathcal{X}_A\big)^T\times\mathcal{X}_S$.

Le problème (2.1) correspond au problème POMDP avec politique *sans mémoire*, ce qui signifie que l'action prise à un certain temps ne dépend que de l'observation courante au lieu de dépendre de l'historique des observations et des actions passées. Les POMDPs sont une généralisation des *Processus de Décision Markovien* (MDPs) où le décideur a accès à l'état courant du système. Alors que les MDPs peuvent être résolus en temps polynomial, le problème POMDP avec politiques sans mémoire est NP-difficile [87]. À moins que P=NP, le fait que le décideur ait seulement accès à une observation partielle rend le problème plus difficile à résoudre.

Une contribution de cette thèse est un *Programme Linéaire en Nombres Entiers* (PLNE), un outil standard de recherche opérationnelle, qui donne une politique optimale du problème POMDP avec politiques sans mémoire. Un tel problème d'optimisation se formule par une fonction objectif linéaire, des contraintes linéaires et des variables entières avec la forme canonique suivante :

$$\begin{aligned}
\min_{\mathbf{x}}\quad & \mathbf{c}^{\mathbf{T}}\mathbf{x}\\
\text{sous les contraintes}\quad & \mathbf{A}\mathbf{x}\leqslant\mathbf{b}\\
& \mathbf{x}\geqslant 0\\
& \mathbf{x}\in\mathbb{Z}^p\times\mathbb{R}^{n-p},
\end{aligned}$$

où $\mathbf{b}$ et $\mathbf{c}$ sont des vecteurs réels et $\mathbf{A}$ une matrice réelle.

Alors que le problème de maximisation d'un MDP peut être résolu par un *programme linéaire* (voir, p. ex., [124]), c-à-d, sans contraintes d'intégrité, notre PLNE résout le problème de maximisation du POMDP (2.1). Ces formulations en nombres entiers peuvent être résolues de manière efficace avec des solveurs commerciaux. Comme ces solveurs utilisent l'algorithme de *séparation et évaluation*[2], il est usuel d'introduire des *inégalités valides* qui améliorent la qualité des bornes obtenues lorsqu'on résout le problème sans les contraintes d'intégrité. Une seconde contribution de cette thèse est un ensemble d'inégalités valides qui aident la résolution de notre PLNE. Plusieurs expérimentations numériques montrent l'efficacité de ces inégalités valides.

Le cadre du POMDP pour le problème de maintenance prédictive avec contraintes de capacité. Il est naturel de supposer que chaque composant se comporte comme un POMDP. Pour chaque composant l'état de dégradation, l'observation partielle et la décision de maintenance (voir figure 2.3) sont respectivement représentés par l'état, l'observation et l'action d'un POMDP. Pour chaque composant, toutes les probabilités conditionnelles $p(o|s)$ et $p(s'|s,a)$ sont calculées à partir des paramètres de la chaîne de Markov cachée. Pour un système de M composants, le problème se divise en M sous-problèmes qui sont *faiblement* couplés par les contraintes de capacité qui affectent les actions prises sur chaque composant. Par exemple, la contrainte de capacité "*le nombre de composants qu'on peut réparer durant une plage de maintenance ne peut dépasser K,*" est modélisée par la contrainte $\sum_{m=1}^{M} a^m \leqslant K$, où a^m représente l'action prise sur le composant m et vaut 1 si le composant m est réparé et 0 sinon. Comme les actions prises sur chaque composant sont couplées, la politique de maintenance doit être considérée sur l'ensemble du système. À chaque pas de temps t, la politique de maintenance est de la forme $\delta^t_{\mathbf{a}|\mathbf{o}}$, où $\mathbf{a} = (a^1, \ldots, a^M)$ et $\mathbf{o} = (o^1, \ldots, o^M)$ représentent respectivement l'action et l'observation du système. Pour chaque composant m, une transition (s, a, s') amène une récompense immédiate

$$r^m(s,a,s') = -\text{Coût}^m(\text{maintenance})\mathbb{1}_{\text{réparer}}(a) - \text{Coût}^m(\text{panne})\mathbb{1}_{\text{panne}}(s'),$$

où $\mathbb{1}_x(y)$ vaut 1 si $x = y$ et 0 sinon.

Si un décideur a accès à l'état de dégradation de chaque composant, alors chaque sous-problème est un problème de MDP et le problème sur le système complet est un ensemble de *MDPs faiblement couplés* [105]. Dans notre cas, le fait que le décideur ait uniquement accès à des observations partielles nécessite d'introduire la notion de *POMDPs faiblement couplés*. En plus de l'aspect partiellement observé, une deuxième difficulté dans la résolution d'un tel problème vient de la *malédiction de la dimension*. En effet, les tailles des espaces grandissent exponentiellement avec le nombre de composants du système, ce qui rend les algorithmes usuels coûteux en ressources informatiques. Pour comprendre cette difficulté, on peut observer qu'encoder une politique de maintenance $\boldsymbol{\delta}$ revient à stocker une table de taille environ égale à $T\prod_{m=1}^{M} |\mathcal{X}_O^m||\mathcal{X}_A^m|$, ce qui est en générale infaisable.

Pour contourner cette difficulté, on introduit la notion de politique *implicite*. À l'inverse des politiques *explicites* qui encodent complètement la politique comme une solution d'un problème d'optimisation, les politiques implicites sont des politiques telles que, pour chaque observation $\mathbf{o}$, le vecteur $(\delta^t_{\mathbf{a}|\mathbf{o}})_{\mathbf{a}}$ est calculé à l'aide d'un problème d'optimisation adapté. Cela a notamment un intérêt pratique parce que le décideur n'a pas besoin de calculer les valeurs des

[2]Branch-and-Bound en anglais

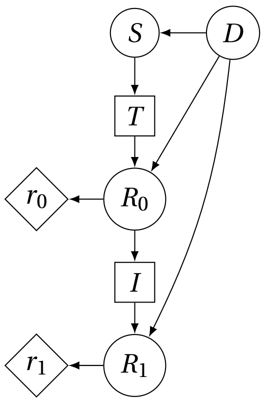
politiques pour toutes les observations. En effet, une fois qu'une observation $\mathbf{o}$ est révélée au temps t, on a seulement besoin de calculer le vecteur $(\delta^t_{\mathbf{a}|\mathbf{o}})_{\mathbf{a}}$. Une autre contribution décrite dans la partie I de cette thèse est une formulation PLNE basée sur celle qu'on a introduit pour le problème de POMDP avec politiques sans mémoire et on l'utilise pour calculer une "bonne" politique implicite pour le problème de POMDPs faiblement couplés. Comme les formulations de Adelman and Mersereau [2] pour les MDPs faiblement couplés, l'avantage principal de notre formulation est de contenir un nombre polynomial de variables et de contraintes, ce qui casse la malédiction de la dimension. Une fois de plus, on améliore notre formulation en ajoutant un ensemble d'inégalités valides pour notre formulation PLNE. De plus, on donne des garanties théoriques sur la valeur optimale de notre formulation PLNE en calculant une *borne inférieure* et une *borne supérieure*.

2.3 Prise de décision avec une incertitude structurée

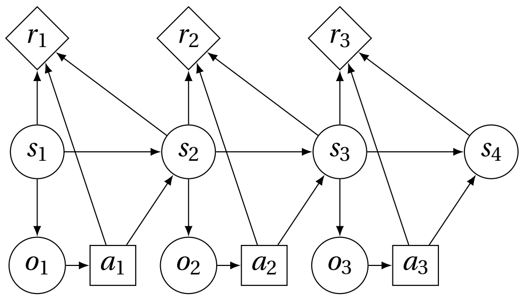
Les MDPs et les POMDPs (voir section 2.2) sont des cas particuliers de problème d'optimisation stochastique discret où la distribution de probabilité des variables aléatoires satisfait une certaine structure qu'on appelle diagramme d'influence. Le diagramme d'influence est un outil flexible qui permet de modéliser un grand ensemble de problèmes d'optimisation stochastique où on suppose que la "nature" est structurée. On introduit le problème d'optimisation dans les diagrammes d'influence à travers un exemple simple.

Cet exemple est représenté en figure 2.4a. On considère un patient qui va consulter un docteur. Le patient souffre d'une maladie D, que le docteur souhaite déterminer. Cette maladie apparaît sur un patient avec la probabilité $\mathbb{P}(D)$, et elle donne des symptômes observables S, qui sont émis aléatoirement avec probabilité $\mathbb{P}(S|D)$. À partir de l'observation des symptômes S, le docteur décide quel traitement T appliquer au patient. Le traitement T et la maladie D donnent une réponse R_0 du patient avec probabilité $\mathbb{P}(R_0|T,D)$, avec une récompense $r_0(R_0)$ indiquant la présence ou non d'effets secondaires indésirables. Ensuite, à partir de cette réponse R_0 le docteur décide si la maladie nécessite une opération médicale plus importante I. On obtient une nouvelle réponse R_1 avec probabilité $\mathbb{P}(R_1|I,D)$, qui indique si l'opération sur la maladie D a été un succès ou non, et on modélise cela avec une récompense $r_1(R_1)$. La maladie D, les symptômes S, les réponses R_0 et R_1, représentent les variables aléatoires sur lesquels le docteur n'a pas de contrôle. Elles sont appelées *variables chances* parce que leurs valeurs sont choisies par la "nature", c-à-d, tout ce sur quoi le docteur n'a pas le contrôle. À l'inverse des variables chances, le traitement T et l'opération médicale I sont représentés par des variables aléatoires qu'on appelle *variables de décision*, dont les valeurs sont choisies par le docteur. Les fonctions r_0 et r_1 sont appelées *fonctions d'utilité* qui dépendent respectivement des réponses R_0 et R_1. L'objectif du docteur est de choisir une stratégie $\boldsymbol{\delta} = \big(\delta_T(T|S), \delta_I(I|R_0)\big)$ qui maximise l'espérance de son utilité $\mathbb{E}_{\boldsymbol{\delta}}\big[r_0(R_0) + r_1(R_1)\big]$ où l'espérance est prise selon la distribution de probabilité $\mathbb{P}_{\boldsymbol{\delta}}(D,S,T,R_0,I,R_1) = \mathbb{P}(D)\mathbb{P}(S|D)\delta_T(T|S)\mathbb{P}(R_0|T,D)\delta_I(I|R_0)\mathbb{P}(R_1|I,D)$.

Ce type de problème peut être encodé graphiquement à l'aide d'un *graphe orienté acyclique*, dont l'ensemble de sommets V contient trois sortes de sommets : les sommets chances (V^{s}), décisions (V^{a}) et utilités (V^{r}) qui correspondent respectivement aux variables chances, de décision et d'utilité. Le diagramme d'influence, qui a été introduit par Howard and Matheson [56], est un graphe orienté acyclique sur cet ensemble de sommets et où il n'y a pas d'arcs sortants



(a) Le problème de décision médicale



(b) Un POMDP avec politique sans mémoire.

Figure 2.4 – Les deux diagrammes d'influence qui modélisent l'exemple de la décision médicale et le POMDP avec politique sans mémoire. Les sommets en forme de cercles, carrés et losanges représentent respectivement les sommets chances, décisions et utilités.

des sommets utilités. La figure 2.4a représente le diagramme d'influence qui modélise le problème de décision médicale qu'on a décrit précédemment. Pour chaque variable chance X_v, représenté par un sommet chance v, on associe une probabilité conditionnelle $\mathbb{P}(X_v|X_{\mathrm{pa}(v)})$ de X_v sachant $X_{\mathrm{pa}(v)}$, où $X_{\mathrm{pa}(v)}$ est un vecteur de variables aléatoires qui correspondent aux parents de v dans le diagramme d'influence, c'est-à-dire l'ensemble de sommets qui ont un arc sortant vers v. Pour chaque fonction utilité r_v, représentée par le sommet utilité v, on associe une fonction déterministe r_v qui à chaque valeur instanciée de $X_{\mathrm{pa}(v)}$ retourne un réel. Les choix du décideur sont modélisés à l'aide d'une *stratégie* $\boldsymbol{\delta} = (\delta_v)_{v \in V^{\mathrm{a}}}$ qui est un vecteur de probabilités conditionnelles $\delta_v(X_v|X_{\mathrm{pa}(v)}) = \mathbb{P}(X_v|X_{\mathrm{pa}(v)})$. Étant donnée une stratégie $\boldsymbol{\delta}$, la distribution de probabilité jointe sur l'ensemble des variables aléatoires s'écrit

$$\mathbb{P}_{\boldsymbol{\delta}}(X_{V^{\mathrm{c}} \cup V^{\mathrm{a}}} = x_{V^{\mathrm{c}} \cup V^{\mathrm{a}}}) = \prod_{v \in V^{\mathrm{c}}} \mathbb{P}(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}) \prod_{v \in V^{\mathrm{a}}} \delta_v(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}). \quad (2.4)$$

Le problème d'optimisation que le décideur cherche à résoudre est le problème du *maximum d'utilité espérée*

$$\max_{\boldsymbol{\delta} \in \Delta} \mathbb{E}_{\boldsymbol{\delta}} \left[\sum_{v \in V^{\mathrm{r}}} r_v(X_{\mathrm{pa}(v)}) \right], \quad (2.5)$$

où Δ est l'ensemble des probabilités conditionnelles possibles pour chaque sommet décision.

On peut observer la capacité de modélisation des diagrammes d'influence sur l'exemple des POMDPs décrit dans la section 2.2. Les variables chances sont les états $(S_t)_{1 \leqslant t \leqslant T+1}$ et les observations $(O_t)_{1 \leqslant t \leqslant T}$, les variables de décision sont les actions $(A_t)_{1 \leqslant t \leqslant T}$, et puis les variables d'utilité sont les récompenses $\big(r(S_t, A_t, S_{t+1})\big)_{1 \leqslant t \leqslant T}$. La distribution de probabilité $\mathbb{P}_{\boldsymbol{\delta}}$ en (2.2) peut être écrite sous la forme (2.4) à l'aide d'un diagramme d'influence représenté sur la figure 2.4b. L'objectif est alors de choisir une politique $\boldsymbol{\delta}$ qui maximise la récompense totale espérée $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\big]$ où l'espérance est prise selon la distribution de probabilité $\mathbb{P}_{\boldsymbol{\delta}}$.

On s'intéresse alors à la question de trouver une stratégie qui résout le problème du maximum d'utilité espérée dans un diagramme d'influence. Ce problème soulève deux questions scientifiques. Premièrement, évaluer une stratégie est déjà difficile. En effet, cela revient à résoudre le *problème d'inférence* dans un diagramme d'influence. Étant donnés une stratégie réalisable $\boldsymbol{\delta}$ et un sous-ensemble de sommets $C \subseteq V$, le problème d'inférence consiste à calculer la probabilité marginale $\mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C)$. Deuxièmement, optimiser sur l'ensemble des stratégies possibles Δ est aussi difficile du fait que le nombre possible de vecteurs de probabilités conditionnelles est très important.

Dans la partie II, on se focalise sur la deuxième question. On généralise les résultats obtenus pour les POMDPs de la section 2.2 à tout problème du maximum d'utilité espérée dans un diagramme d'influence. Nos résultats sont basés sur plusieurs outils de la *théorie des modèles graphiques probabilistes* et la *programmation mathématique en nombres entiers*. Une première contribution décrite dans la partie II est un PLNE qui donne une stratégie optimale du problème du maximum d'utilité espérée dans un diagramme d'influence. Alors que les solutions proposées dans la littérature sont basées sur des algorithmes de type *programmation dynamique* sur le graphe (voir, p. ex., [76, 81, 89]), notre approche est basée sur la programmation mathématique. Une seconde contribution est un ensemble d'inégalités valides pour notre formulation en nombres entiers. Une troisième contribution est une caractérisation du cas “simple” de diagramme d'influence, c-à-d, les diagrammes d'influence tels que le problème du maximum d'utilité espérée est plus facile à résoudre. Une partie des résultats de la partie II a été publiée dans Parmentier et al. [117]. Plusieurs nouveaux résultats ont été ajoutés après la publication de cet article.

2.4 Une méthodologie statistique pour le problème de maintenance des avions

On souhaite utiliser notre politique pour les POMDPs faiblement couplés, mentionnée dans la section 2.2, pour nôtre problème appliqué. Cependant, modéliser un problème de maintenance d'un système réel comme un problème de maintenance prédictive avec contraintes de capacité comme dans la section 2.1 n'est pas immédiat. En particulier, c'est le cas de notre problème de maintenance des avions chez Air France. En effet, les données disponibles à chaque plage de maintenance sont des données brutes enregistrées par les capteurs sur les équipements des avions et correspondent aux valeurs de signaux enregistrés durant les vols. Un exemple de signal enregistré par un capteur est l'évolution de la pression au sein d'un équipement échantillonnée à 1Hz durant un vol. C'est pourquoi notre base de données contient plusieurs années de données brutes et quelques dates de panne. Par conséquent, exprimer le problème de maintenance des avions comme un problème de maintenance prédictive avec contraintes de capacité nécessite le calcul des paramètres des chaînes de Markov cachées qui modélisent les processus de détérioration des équipements à partir de cette base de données. Nous proposons une méthodologie statistique préliminaire qui permet de transformer le problème de maintenance des avions (figure 2.1) en un problème de maintenance prédictive avec contraintes de capacité (figure 2.3).

Cela soulève trois questions pratiques. Premièrement, dans le problème de la maintenance prédictive avec contraintes de capacité, le décideur a accès à un indicateur discret comme observation partielle, alors que les observations du problème de la maintenance des avions correspondent à des données brutes, qui sont continues et en grande dimension. Il est donc nécessaire de développer une méthodologie qui permette de transformer les valeurs des signaux en indicateurs discrets.

La deuxième question concerne la phase d'*apprentissage*. Sur la base des données historiques, l'objectif est d'estimer les paramètres qui définissent complètement le modèle de Markov caché pour chaque équipement, et qui sont nécessaires comme entrées du problème de maintenance prédictive avec les contraintes de capacité.

La troisième question concerne l'*interprétabilité* de l'approche. En effet, même si elles n'ont pas d'impact immédiat sur la sécurité des vols[3], les décisions de maintenance que nous suggérons ne sont pas bénignes. Les opérations de maintenance telles que le démontage du train d'atterrissage d'un avion long-courrier sont coûteuses et prennent du temps. Et si une panne se produit, Air France devra annuler les prochains vols effectués par l'avion, ce qui est encore plus coûteux. Alors, si nous voulons que notre modèle soit utilisé dans la pratique, les prédictions du modèle statistique, le modèle de Markov caché, ainsi que les décisions de maintenance résultant d'une politique de maintenance, doivent être fiables du point de vue des ingénieurs de maintenance. Une difficulté supplémentaire sur ce point est le fait que la base données que nous considérons contient un faible nombre de pannes du fait qu'Air France essaie de les éviter autant que possible et que nous disposons d'un faible nombre de données brutes correspondant à un comportement défaillant d'un équipement avant la panne. Par conséquent, nous ne pouvons pas valider le modèle statistique à l'aide de validations expérimentales.

Notre approche est décrite dans le chapitre 10 et aborde ces questions en combinant quatre étapes qui utilisent plusieurs outils statistiques. Premièrement, comme les données brutes que nous traitons sont en grande dimension et bruitées, nous transformons les séries temporelles enregistrées pendant un vol par un nombre raisonnable d'*attributs* pertinents. Par exemple, si un capteur enregistre l'évolution de la pression dans un équipement pendant un vol, un exemple d'attribut est la moyenne ou l'écart-type de la pression sur la durée du vol. Ensuite, nous apprenons les paramètres d'un *modèle de Markov caché gaussien* qui prédit l'évolution du vecteur des attributs. Un tel modèle est un modèle de Markov caché où les observations sont continues et sont émises selon une distribution de probabilité gaussienne. Troisièmement, nous transformons ce modèle de Markov caché gaussien en un modèle de Markov caché avec observations discrètes en utilisant un *arbre de décision*. Il prend un vecteur d'observations continues en entrée et renvoie un indicateur discret. Le mécanisme au sein d'un arbre de décision est une combinaison de règles binaires telles que “est-ce que la pression moyenne est supérieure à 20 bars ?”, qui conduit à un indicateur discret, dont les valeurs correspondent aux différents niveaux de risques de panne.

Ces règles binaires peuvent être facilement comprises et validées par les ingénieurs de la maintenance, ce qui répond au troisième problème. Comme cet indicateur est discret, il répond également à la première question. Enfin, à partir du modèle de Markov caché gaussien et de l'arbre de décision, nous calculons les paramètres du modèle de Markov caché avec indicateurs discrets, ce qui répond au deuxième problème. L'utilisation de cette approche permet de considérer le problème de maintenance des avions chez Air France comme un problème de

[3]Un équipement critique tombe en panne lorsqu'il atteint un seuil conservatif fixé par les agences de régulation, et au-delà duquel les avions ne sont plus autorisés à décoller.

maintenance prédictive avec contraintes de capacité. Les expérimentations numériques montrent que notre politique de maintenance donne de “bons” résultats numériques comparés à ceux obtenus en utilisant la politique de maintenance d’Air France.

Part I

Integer programming for predictive maintenance

3 Predictive maintenance with capacity constraints

This chapter formalizes the predictive maintenance problem with capacity constraints as a Partially Observable Markov Decision Process (POMDP). We recall the context of such a problem. We consider a system composed of multiple deteriorating components, which deteriorates over time. We suppose that there are scheduled decision times, or maintenance slots, where a decision maker decides which components to maintain. Due to the maintenance capacity, only a limited number of components can be maintained. At each maintenance slot, the decision maker has access to a discrete observation of the system. His decision is modeled using a maintenance policy, which maps an observation to a decision. The goal of the decision maker is to find a maintenance policy minimizing the costs expressed as the sum of the expected failure costs and the expected maintenance costs. We use this objective function because in practice the airline knows the failure costs and the maintenance costs.

The POMDP is a natural way to see the predictive maintenance problem [24]. However, when the system has several components, the size of its state space becomes exponential in the number of components, which makes the corresponding POMDP intractable. This issue is known as the *curse of dimensionality* [12]. To address this issue, we exploit the fact that the system can be decomposed into several components. To do so, we formalize the predictive maintenance problem with capacity constraints on a system with multiple components and capacity constraints using *weakly coupled POMDP* that we introduce and which is analogs of the *weakly coupled dynamic programs* of Adelman and Mersereau [2].[1] In this formalization, we assume that the observations are discrete and all the model parameters are known. In Chapter 10, we will use the weakly coupled POMDP to address the concrete predictive maintenance problem of Air France and this assumption will be discussed.

Chapter 3 is organized as follows.

- Section 3.1 recalls the necessary background on POMDPs.
- Section 3.2 introduces the notion of weakly coupled POMDP as a special case of POMDP.
- Section 3.3 describes how we formalize the predictive maintenance problem with capacity constraints using a weakly coupled POMDP.
- Section 3.4 shows examples that can be formalized as a weakly coupled POMDP.
- Finally, Section 3.5 contains bibliographical remarks on maintenance problems, and the use of POMDPs for maintenance optimization.

[1] Weakly coupled dynamic programs are also called weakly coupled Markov decision processes in [15].

3.1 Background on POMDP

In this section, we recall the POMDP problem and the POMDP problem with *memoryless* policies. In Section 3.3, we explain why we choose a memoryless policy for the predictive maintenance problem and several numerical experiments in Section 4.5 support our choice.

3.1.1 POMDP parameters

We recall here the definition of a POMDP. A POMDP is a multi-stage stochastic optimization problem defined as follows. It models on a horizon T in $\mathbb{Z}_+$ the evolution of a system. At each time t in $[T]$, the system is in a random state S_t, which belongs to a finite state space $\mathcal{X}_S$. The system starts in state s in $\mathcal{X}_S$ with probability $p(s) := \mathbb{P}(S_1 = s)$. At time t, the decision maker does not have access to S_t, but observes O_t, whose value belongs to a finite state $\mathcal{X}_O$. When the system is in state $S_t = s$, it emits an observation $O_t = o$ with probability $\mathbb{P}(O_t = o|S_t = s) := p(o|s)$. Then, the decision maker takes an action A_t, which belongs to a finite space $\mathcal{X}_A$. Given an action $A_t = a$, the system transits from state $S_t = s$ to state $S_{t+1} = s'$ with probability $\mathbb{P}\big(S_{t+1} = s'|S_t = s, A_t = a\big) := p(s'|s,a)$, and the decision maker receives the immediate reward $r(s,a,s')$, where the reward function is defined as a real valued function $r\colon \mathcal{X}_S \times \mathcal{X}_A \times \mathcal{X}_S \to \mathbb{R}$, which we will also view as a vector $\mathbf{r} = \big(r(s,a,s')\big) \in \mathbb{R}^{\mathcal{X}_S \times \mathcal{X}_A \times \mathcal{X}_S}$.

We denote by $\mathfrak{p}$ the vector of probabilities $\mathfrak{p} = \big(p(s), p(o|s), p(s'|s,a)\big)_{\substack{s,s' \in \mathcal{X}_S, o \in \mathcal{X}_O \\ a \in \mathcal{X}_A}}$. A POMDP is parametrized by the fifth tuple $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$.

3.1.2 POMDP problem

Let $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$ be a POMDP. Given a finite horizon $T \in \mathbb{Z}_+$, the choices made by the decision maker are modeled using a policy $\boldsymbol{\delta} = (\delta^1, \dots, \delta^T)$, where δ^t is the conditional probability distribution of taking action A_t at time t given the history of observations and actions $H_t = (O_1, A_1, \dots, A_{t-1}, O_t)$ in $\mathcal{X}_H^t := (\mathcal{X}_O \times \mathcal{X}_A)^t \times \mathcal{X}_O$, i.e.,

$$\delta^t_{a|h} := \mathbb{P}(A_t = a | H_t = h),$$

for any a in $\mathcal{X}_A$ and h in $\mathcal{X}_H^t$. We denote by Δ_{his} the set of policies

$$\Delta_{\text{his}} = \left\{ \boldsymbol{\delta} \in \mathbb{R}^{\sum_{t=1}^T \mathcal{X}_H^t \times \mathcal{X}_A} \colon \sum_{a \in \mathcal{X}_A} \delta^t_{a|h} = 1 \text{ and } \delta^t_{a|h} \geqslant 0, \forall h \in \mathcal{X}_H^t, a \in \mathcal{X}_A, t \in [T] \right\}.$$

In Δ_{his}, "his" refers to policies that take into account the history of observations and actions by opposition to memoryless policies which will be introduced below. A policy $\boldsymbol{\delta} \in \Delta_{\text{his}}$ leads to the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ on $(\mathcal{X}_S \times \mathcal{X}_O \times \mathcal{X}_A)^T \times \mathcal{X}_S$ such that

$$\mathbb{P}_{\boldsymbol{\delta}}\big((S_t = s_t, O_t = o_t, A_t = a_t)_{1 \leqslant t \leqslant T}, S_{T+1} = s_{T+1}\big) = p(s_1) \prod_{t=1}^T p(o_t|s_t) p(s_{t+1}|s_t, a_t) \delta^t_{a_t|h_t},$$

where $h_t = (s_1, o_1, \dots, a_{t-1}, o_t)$. We denote by $\mathbb{E}_{\boldsymbol{\delta}}$ the expectation according to $\mathbb{P}_{\boldsymbol{\delta}}$. The goal of the decision maker is to find a policy $\boldsymbol{\delta}$ in Δ_{his} maximizing the expected total reward over the finite horizon T. The POMDP problem is exactly the following:

$$\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{his}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right] \tag{P$_{\mathrm{his}}$}$$

It is known that (P$_{\mathrm{his}}$) is PSPACE-hard [113].

3.1.3 POMDP problem with memoryless policies

Let $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$ be a POMDP. Given a finite horizon $T \in \mathbb{Z}_+$, a memoryless policy is a vector $\boldsymbol{\delta} = (\delta^1, \dots, \delta^T)$, where δ^t is the conditional probability distribution at time t of action A_t given observation O_t, i.e.,

$$\delta^t_{a|o} := \mathbb{P}(A_t = a | O_t = o)$$

for any a in $\mathcal{X}_A$ and o in $\mathcal{X}_O$. Such policies are said memoryless because the choice of A_t only depends on the current observation O_t, in contrast with the history of observations and actions H_t. We denote by Δ_{ml} the set of memoryless policies

$$\Delta_{\mathrm{ml}} = \left\{\boldsymbol{\delta} \in \mathbb{R}^{T\times\mathcal{X}_A\times\mathcal{X}_O} \colon \sum_{a\in\mathcal{X}_A} \delta^t_{a|o} = 1 \text{ and } \delta^t_{a|o} \geqslant 0, \forall o \in \mathcal{X}_O, a \in \mathcal{X}_A\right\}.$$

In Δ_{ml}, "ml" refers to memoryless. A policy $\boldsymbol{\delta} \in \Delta_{\mathrm{ml}}$ leads to the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ on $(\mathcal{X}_S \times \mathcal{X}_O \times \mathcal{X}_A)^T \times \mathcal{X}_S$ such that

$$\mathbb{P}_{\boldsymbol{\delta}}\big((S_t = s_t, O_t = o_t, A_t = a_t)_{1\leqslant t\leqslant T}, S_{T+1} = s_{T+1}\big) = p(s_1)\prod_{t=1}^{T} p(o_t|s_t)p(s_{t+1}|s_t, a_t)\delta^t_{a_t|o_t}. \tag{3.1}$$

We denote by $\mathbb{E}_{\boldsymbol{\delta}}$ the expectation according to $\mathbb{P}_{\boldsymbol{\delta}}$. The goal of the decision maker is to find a memoryless policy $\boldsymbol{\delta}$ in Δ_{ml} maximizing the expected total reward over the finite horizon T. The POMDP problem with memoryless policy is exactly the following:

$$\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right] \tag{P$_{\mathrm{ml}}$}$$

The definition of Δ_{ml} ensures that $\Delta_{\mathrm{ml}} \subseteq \Delta_{\mathrm{his}}$. The policy is said to be memoryless because the decision maker takes its decision given the current observation instead of the full history of observations and actions. To clarify the distinction between Δ_{ml} and Δ_{his}, we say that a policy belonging to Δ_{his} is a history-dependent policy. It is known that (P$_{\mathrm{ml}}$) is NP-hard [87]. In Section 4.5, we provide numerical experiments showing that memoryless policies perform well on different kind of problems modeled as a POMDP, especially the maintenance problems.

3.2 Weakly coupled POMDP

Like MDPs, the POMDPs suffer from the curse of dimensionality. We can observe this phenomenon when we consider systems that consist of a collection of smaller subsystems or components. It is the case of the predictive maintenance problem we want to formalize within the POMDP framework. In order to catch and leverage the specific structure of systems with several components, we introduce a special case of POMDP, the weakly coupled POMDP. We give its formal definition in this paragraph.

A weakly coupled POMDP models a system composed of M components, each of them evolving independently as a POMDP. Let S_t^m and O_t^m be random variables that represent respectively the state and the observation of component m at time t, and that belong respectively to the state space $\mathcal{X}_S^m$ and the observation space $\mathcal{X}_O^m$ of component m. Each component is assumed to evolve individually as a POMDP. We denote by $\mathfrak{p}^m$ and $\mathbf{r}^m$ respectively the probability distributions and the immediate reward functions of component m. We denote by $\mathbf{S}_t = \left(S_t^1, \ldots, S_t^M\right)$ and $\mathbf{O}_t = \left(O_t^1, \ldots, O_t^M\right)$ the state and the observation of the full system at time t, which lie respectively in the *state space* $\mathcal{X}_S = \mathcal{X}_S^1 \times \cdots \times \mathcal{X}_S^M$ and the *observation space* $\mathcal{X}_O = \mathcal{X}_O^1 \times \cdots \times \mathcal{X}_O^M$. The spaces $\mathcal{X}_S$ and $\mathcal{X}_O$ represent the state space and the observation space of the full system. We assume that the action space $\mathcal{X}_A$ can be written

$$\mathcal{X}_A = \left\{ \mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M \colon \sum_{m=1}^M \mathbf{D}^m(a^m) \leqslant \mathbf{b} \right\}, \tag{3.2}$$

where $\mathcal{X}_A^m$ is the individual action space of component m, and $\mathbf{D}^m : \mathcal{X}_A^m \to \mathbb{R}^q$ is a given function for each component m in $[M]$, and $\mathbf{b} \in \mathbb{R}^q$ is a given vector for some finite integer q. Without loss of generality, we assume that $\mathbf{b} \geqslant 0$ in the remaining of this thesis[2].

Each component is assumed to evolve independently, hence the joint probability of emission factorizes as

$$\mathbb{P}(\mathbf{O}_t = \mathbf{o} | \mathbf{S}_t = \mathbf{s}) = \prod_{m=1}^M p^m(o^m | s^m), \tag{3.3}$$

and the joint probability of transition factorize as

$$\mathbb{P}(\mathbf{S}_1 = \mathbf{s}) = \prod_{m=1}^M p^m(s^m) \quad \text{and} \quad \mathbb{P}(\mathbf{S}_{t+1} = \mathbf{s}' | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}) = \prod_{m=1}^M p^m(s'^m | s^m, a^m), \tag{3.4}$$

for all t in $[T]$. In addition, the reward is assumed to decompose additively

$$r(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \sum_{m=1}^M r^m(s^m, a^m, s'^m). \tag{3.5}$$

Hence, the weakly coupled POMDP problem with memoryless policies is the following:

$$\max_{\delta \in \Delta_{\mathrm{ml}}} \mathbb{E}_{\delta} \left[\sum_{t=1}^T r(\mathbf{S}_t, \mathbf{A}_t, \mathbf{S}_{t+1}) \right] \tag{$\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$}$$

where the expectation is taken according to $\mathbb{P}_{\delta}$ defined in (3.1). Note that unless $\mathcal{X}_A = \emptyset$ there always exists a feasible policy of ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). A weakly coupled POMDP is fully parametrized by $\left((\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m \in [M]}, \mathbf{b}\right)$.

($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$) is equivalent to ($\mathrm{P}_{\mathrm{ml}}$) on the state space $\mathcal{X}_S = \mathcal{X}_S^1 \times \cdots \times \mathcal{X}_S^M$, the observation space $\mathcal{X}_O =$

[2]It suffices to set $\mathbf{D}'^m := \mathbf{D}^m - \frac{k}{M}$, $\mathbf{b}' := \mathbf{b} - k$ where $k = \min_{i \in [q]} b_i$ and b_i indicates the i-th coordinate of vector $\mathbf{b}$, and $\tilde{\mathcal{X}}_A = \left\{ \mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M \colon \sum_{m=1}^M \mathbf{D}'^m(a^m) \leqslant \mathbf{b}' \right\}$. Then, it follows that $\mathcal{X}_A = \tilde{\mathcal{X}}_A$.

$\mathcal{X}_O^1 \times \cdots \times \mathcal{X}_O^M$, and the action space $\mathcal{X}_A$ defined in (3.2). This notion of weakly coupled POMDP is an extension of the weakly coupled dynamic program introduced by Adelman and Mersereau [2] to the case where the decision maker has only access to a partial observation of the system state. The definition (3.2) of the action space for weakly coupled POMDP is exactly the same as Bertsimas and Mišić [15, Eq. (7)] for weakly coupled dynamic programs. Note that we can also defined the weakly coupled POMDP problem with history-dependent policies ($\mathrm{P}_{\mathrm{his}}^{\mathrm{wc}}$).

Remark 1. In the definition of POMDP, we could have considered a variant where the observation O_t may depend on A_{t-1} given S_t and the emission probability distribution becomes $\mathbb{P}(O_t = o | A_{t-1} = a', S_t = s) := p(o|a', s)$. All the mathematical programming formulations and theoretical results in this thesis can be extended to this case. We choose to consider the case above to lighten the notation. △

Remark 2. In many stochastic problems, the action space does not decompose along the components, i.e., we can no longer assume that $\mathcal{X}_A \subseteq \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M$. It turns out that, using a transformation, we can always convert such a problem into a weakly coupled POMDP. This transformation is similar to the one introduced by Bertsimas and Mišić [15, Sec 4.3]. We set the individual action spaces $\mathcal{X}_A^m = \mathcal{X}_A$ for each component m in $[M]$. For any $(a^1, \ldots, a^M) \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M$, we enforce the following linking constraints $a^m = a^{m+1}$ for all m in $[M]$. Therefore, the action space can be written $\left\{\mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M \colon a^m = a^{m+1}, \ \forall m \in [M-1]\right\}$, which has the requested form (3.2). △

3.3 Formalizing the predictive maintenance problem with capacity constraints

In this section we formalize the predictive maintenance problem with capacity constraints using a weakly coupled POMDP. It requires to define the parameters $\left((\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m \in [M]}, \mathbf{b}\right)$.

The system is composed of M components. At each maintenance slot $t \in [T]$, the decision maker receives an observation O_t^m from component m, which belongs to a finite space $\mathcal{X}_O^m$. We model the degradation of component m using a state S_t^m, which belongs to a finite state space $\mathcal{X}_S^m$ and is not observed by the decision maker. We assume that there is a *failure state* $s^{m,F}$ in $\mathcal{X}_S^m$ for each component m in $[M]$, corresponding to its most critical degradation state. We denote by $\mathbf{S}_t = \left(S_t^1, \ldots, S_t^M\right)$ and $\mathbf{O}_t = \left(O_t^1, \ldots, O_t^M\right)$ the state and observation of the full system at time t, which lie respectively in the state space $\mathcal{X}_S = \mathcal{X}_S^1 \times \cdots \times \mathcal{X}_S^M$ and the observation space $\mathcal{X}_O = \mathcal{X}_O^1 \times \cdots \times \mathcal{X}_O^M$.

Component m starts in state s with probability $p^m(s)$. At each time t, component m is in state $S_t^m = s$, and it emits an observation $O_t^m = o$ with probability $p^m(o|s)$. Then, the decision maker takes an action $\mathbf{A}_t = \left(A_t^1, \ldots, A_t^M\right)$ in a finite space $\mathcal{X}_A$, where A_t^m is a binary variable equal to 1 when component m is maintained. At each maintenance slot, the decision maker can maintain at most K components. Hence, we write the action space $\mathcal{X}_A$ as follows

$$\mathcal{X}_A = \left\{\mathbf{a} = (a^1, \ldots, a^M) \in \{0,1\}^M \colon \sum_{m=1}^{M} a^m \leqslant K\right\}. \tag{3.6}$$

Therefore, $\mathcal{X}_A$ contains only one scalar constraint ($q = 1$) and satisfies (3.2) by setting $D^m(a^m) =$

a^m for every $a^m \in \{0,1\}$ and $m \in [M]$, and $b = K$. We assume that each component m evolves independently from state $S_t^m = s$ to state $S_{t+1}^m = s'$ with probability $p^m(s'|s,a)$, and the decision maker receives reward $r^m(s,a,s')$. In addition, we assume that when a component is maintained, it behaves like a new one, i.e.,

$$p^m(s'|s,1) = p^m(s'), \tag{3.7}$$

for any s, s' in $\mathcal{X}_S^m$, and the conditional probabilities factorize as (3.3) and (3.4). Each component has a maintenance cost C_R^m and a failure cost C_F^m at each component m. The individual immediate reward function can be written

$$r^m(s,a,s') = -\mathbb{1}_{s_F^m}(s')C_F^m - \mathbb{1}_1(a)C_R^m, \tag{3.8}$$

for any $s, s' \in \mathcal{X}_S^m$ and $a \in \mathcal{X}_A^m$. We assume that the reward decomposes additively as (3.5). Since our motivation was to find the right framework to formalize the maintenance problem, it would have been more natural to define ($\mathrm{P_{ml}}$) as a minimization problem. However, we choose to stick to the practice of using rewards, which is common in the literature on POMDPs.

We model the choices of the decision maker using a memoryless policy. The interest of memoryless policies is threefold. First, maintenance technicians can easily apply them in practice. Second, even if finding an optimal memoryless policy remains NP-hard [87], the resulting optimization problem is more tractable. Third, even if in the general case memoryless policies are not optimal, there is a broad class of systems for which memoryless policies perform well [10, 84] and we provide numerical experiments in Section 4.5 on instances from the literature showing that it is the case for maintenance problems.

Given a finite horizon T, the predictive maintenance problem with capacity constraints consists in finding a policy in Δ_{ml} that solves ($\mathrm{P_{ml}^{wc}}$) with $\mathcal{X}_A$ defined in (3.2), $(\mathfrak{p}^m)_{m\in[M]}$ satisfying (3.7), and $(\mathbf{r}^m)_{m\in[M]}$ defined in (3.8).

Remark 3. In fact, we could have modeled the predictive maintenance problem with capacity constraints as a weakly coupled POMDP over an infinite horizon with a discounted reward. Indeed, in practice, the predictive maintenance problem consists in planning the maintenances over a large horizon, which can be modeled by an infinite horizon planning problem. In this case, the predictive maintenance problem with capacity constraints consists in finding a policy that maximizes the total expected discounted reward over infinite horizon $\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{his}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r(\mathbf{S}_t,\mathbf{A}_t,\mathbf{S}_{t+1})\right]$ where $\gamma < 1$ is a discount factor. However, in practice Air France would like to modify the maintenance planning over a short horizon without discounting. Since the observations arrive dynamically, Air France would like to modify its planning in an "on-line" manner due to unexpected changes detected in the behavior of an equipment. In addition, as we will explain in Chapter 10, the parameters of the POMDPs are estimated through a statistical methodology and sometime the resulting long term predictions can be inaccurate. It pushes us to consider POMDP with finite horizon. This choice is supported by Walraven and Spaan [158, Section 3.1], which showed that casting a finite horizon problem with infinite horizon problem with discounting can lead to widely suboptimal policies. △

3.4 Examples modeled as a weakly coupled POMDP

In this section we describe several multi-stage stochastic optimization problems that can be formalized as a weakly coupled POMDP. More examples of POMDPs applications can be found for instance in the survey of Cassandra [24].

Example 1. *Multi-armed and Restless Bandit* problems are classical resource allocation problems where there are several arms, each of them evolving independently as a MDP, and at each time the decision maker has to activate a subset of arms so as to maximize its expected discounted reward over infinite horizon. We can consider *regular* multi-armed bandit problem, where only the activated arm states transit randomly and give an immediate reward, or the *restless* multi-armed bandit problem, where all the arm states transit randomly and give an immediate reward. When the decision maker has only access to a partial observation on each arm instead of the arm state, the problem becomes a partially observable multi-armed bandit problem [79]. In this case, each arm evolves individually and independently as a POMDP. Such a problem enables to model practical applications such as clinical trials. In this setting, each component represents a medical treatment and activating a component corresponds to testing the treatment. The state of a medical treatment corresponds to its efficiency and the observation corresponds to a noisy measure of the efficiency of a medical treatment.

We can formalize the partially observable multi-armed bandit problem within our weakly coupled POMDP framework. Let M be the number of arms. At each time t, the decision maker has to activate $K < M$ arms. Since each component evolves as POMDP, we use the same notation to represent the state and the observation of Section 3.2. We define the individual action space $\mathcal{X}_A^m = \{0,1\}$ of arm m and the full action space is

$$\mathcal{X}_A = \left\{\mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M : \sum_{m=1}^{M} a^m = K\right\}.$$

In the case of the regular bandit problems, the immediate reward of component m satisfies $r^m(s,0,s') = 0$, and the transition probabilities satisfy $p^m(s'|s,0)$ equals 1 if $s = s'$ and 0 otherwise, for every $s, s' \in \mathcal{X}_S^m$. The goal of the decision maker is to find a policy $\boldsymbol{\delta}$ in Δ_{ml} (or Δ_{his}) maximizing the total expected discounted reward over infinite horizon $\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(\mathbf{S}_t, \mathbf{A}_t, \mathbf{S}_{t+1})\right]$, where T is a finite horizon.

Note that the the predictive maintenance problem with capacity constraints can also be modeled as a restless partially observable multi-armed bandit problem. Indeed, it suffices to add a fictive component $M+1$ corresponding to performing no action, with no reward, and a single state coinciding with a single observation. Then, the action space defined in (3.6) can be written $\mathcal{X}_A = \left\{\mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M : \sum_{m=1}^{M+1} a^m = K\right\}$. $\triangle$

Example 2. Consider a supplier that delivers a product to M stores. At each time t, we denote by S_t^m the inventory level of store m. Unfortunately, due to "inventory records inaccuracy" [103] from various uncertainties, the supplier does not observe directly this inventory level. He has instead only access to a noisy observation O_t^m of the inventory level of store m. We assume that the inventory level of store m has a known limited capacity C^m. Hence, we set $\mathcal{X}_S^m := \{0, \ldots, C^m\}$. Then $O_t^m = o$ is an noisy observation of the current inventory level, whose value belongs to $\mathcal{X}_O^m := \mathcal{X}_S^m$ and is randomly emitted given a current state $S_t^m = s$ according to a known probability $p^m(o|s) = \mathbb{P}(O_t^m = o|S_t^m = s)$. At each time, the supplier has to decide the quantity to produce and to deliver automatically to each store. We denote by A_t^m the delivered quantity of product to store m, which belongs to the individual action space $\mathcal{X}_A^m := \mathcal{X}_S^m$. The production has to satisfy resource constraints (raw materials, staff, etc.). Hence, the set of feasible actions has the form

$$\mathcal{X}_A := \left\{ \mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M : \sum_{m=1}^{M} h^m a^m \leqslant H \right\},$$

where h^m is the given number of resources used per unit produced and delivered for store m and H is the given available amount of resource. The quantity of products in store m cannot exceed capacity C^m. Hence, the quantity $\max(S_t^m + A_t^m - C^m, 0)$ is wasted and it induces a waste cost.

We denote by D_t^m the random variable representing the demand at store m between time t and $t+1$. The vector of demand is exogenous and independent identically distributed in each store with a known probability distribution $\mathbb{P}_D^m$ for store m. The inventory level of store m follows the dynamic

$$S_{t+1}^m = \max\left(\min\left(S_t^m + A_t^m, C^m\right) - D_t^m, 0\right),$$

which gives the transition probability distribution $\mathbb{P}(S_{t+1}|S_t, A_t)$. Now we can define the immediate reward function

$$r^m(s,a,s') = \text{price}^m(s+a-s') - \text{waste}^m \max\left(s + a - C^m, 0\right) - \text{shortage}^m \mathbb{E}_{\mathbb{P}_D^m}\left[\max\left(D^m - (s+a), 0\right)\right],$$

where price^m is the selling price per unit, waste^{m} is the wastage cost per unit and shortage^m is the shortage cost per unit. It leads us to model this problem as a weakly coupled POMDP. The goal of the supplier is to find a policy $\boldsymbol{\delta}$ in Δ_{ml} (or Δ_{his}) maximizing the total expected reward over a finite horizon T. This example has been introduced by Kleywegt et al. [74] for fully observable inventory levels and Mersereau [103] justifies the relevance of the POMDP framework for the stochastic inventory control problem. △

Example 3. Consider a nurse assignment problem for home health care. A medical center follows M patients at home on a daily basis over a given period of time T. On day t, we denote by S_t^m the health state of patient m, whose value belongs to a finite state space $\mathcal{X}_S^m$. The medical center does not directly observe the health state of each patient. However, at each time t, the medical center has access to a partial observation O_t^m corresponding to a signal sent by a machine which diagnoses patient m. We assume that this signal is discrete and noisy. Hence, $\mathcal{X}_O^m$ is a finite space and an observation o is randomly emitted given a state $s \in \mathcal{X}_S^m$ according to the probability $p^m(o|s)$. At each time, the medical center has to assign nurses to patient. There are K_1 available nurses with skill 1 and K_2 available nurses with skill 2. On day t, we denote by A_t^m the action taken by the medical center on patient m, whose value belongs to $\mathcal{X}_A^m = \{0,1,2,3\}$

and the following meaning.

$$A_t^m = \begin{cases} 0 & \text{if no nurse is sent to patient } m \\ 1 & \text{if a nurse with skill 1 is sent to patient } m \\ 2 & \text{if a nurse with skill 2 is sent to patient } m \\ 3 & \text{if two nurses, one with each skill, are sent to patient } m \end{cases}$$

Depending on the skill of the nurses sent to patient, the health state of each patient evolves randomly according to a transition probability $p^m(s'|s,a)$, for any $s,s' \in \mathcal{X}_S^m$, $a \in \mathcal{X}_A^m$ and $m \in [M]$. Hence, the set of feasible actions has the form

$$\mathcal{X}_A = \left\{ \mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M \colon \sum_{m=1}^M \mathbb{1}_1(a^m) + \mathbb{1}_3(a^m) \leqslant K_1 \text{ and } \sum_{m=1}^M \mathbb{1}_2(a^m) + \mathbb{1}_3(a^m) \leqslant K_2 \right\}.$$

Now we can define the immediate reward function

$$r^m(s,a,s') = -\mathrm{cost}_1^{\mathrm{m}}(\mathbb{1}_1(\mathrm{a}) + \mathbb{1}_3(\mathrm{a})) - \mathrm{cost}_2^{\mathrm{m}}(\mathbb{1}_2(\mathrm{a}) + \mathbb{1}_3(\mathrm{a})) - \mathrm{emergency}^{\mathrm{m}} \mathbb{1}_{\mathrm{s}_{\mathrm{critic}}^{\mathrm{m}}}(\mathrm{s}'),$$

where $\mathrm{cost}_\mathrm{i}^\mathrm{m}$ is the cost induced by sending a nurse with skill $i \in \{1,2\}$ to patient m, s_{critic}^m is the critical health state of patient m and $\mathrm{emergency}^\mathrm{m}$ is the cost induced by an emergency because patient m reaches its critical health state. It leads us to model this problem as a weakly coupled POMDP. The goal of the medical center is to find a policy $\boldsymbol{\delta}$ in Δ_{ml} (or Δ_{his}) maximizing its total expected reward over a finite horizon T. △

3.5 Bibliographical remarks

In the past decades, the optimization in maintenance problems of deteriorating systems have been widely studied in academic research. For a complete description of the various types of maintenance problems, see for instance the surveys of Alaswad and Xiang [3], Valdez-Flores and Feldman [154], Wang [159]. The predictive maintenance problem with capacity constraints belongs to the big family of multistage stochastic optimization problems. Since describing such problems is beyond the scope of this thesis, we refer the interested reader to the tutorial of Powell [122].

Partially Observable Markov Decision Processes. The POMDP framework can be adapted for a wide range of various applications such as maintenance problems [42] or clinical decision making [37]. More decision making problems which can be modeled as a POMDP are described in the survey by Cassandra [24]. Modeling a deteriorating system using partially observable continuous time Markov chains has been done in the literature [70, 71, 90, 91]. Kim et al. [72] showed the efficiency of an optimal policy of such a formalization in the mining industry. In all these works, the Markov chains have at most three states. Many scientific works show that the use of POMDPs to model a maintenance problem gives promising results in practice [44, 69]. In particular, it is showed that modeling the maintenance problem as a POMDP instead of a MDP leads to lower maintenance costs.

Memoryless policies. As mentioned in Section 3.3, in addition to their practical advantages (tractability and easy to apply) the memoryless policies perform well on a broad class of systems in practice [10, 84, 106, 164]. It has been even proved that for *contextual MDPs*, which are special cases of POMDPs, there always exists an optimal policy that is memoryless [77]. On the other hand, Littman [87] identifies some pathological cases where the memoryless policies do not work well. Memoryless policies have received much interest with the increasing popularity of reinforcement learning [8, 9, 59, 143]. Indeed, in this context the decision maker does not have access to the parameters $\mathfrak{p}$ and $\mathbf{r}$. Hence, he has to optimize his policy and estimate the parameters at the same time. Azizzadenesheli et al. [7] explain why in this context the memoryless policies are of interest and especially the *stochastic memoryless* policies, which map an observation to a probability distribution over the set of decisions, in opposition to *deterministic memoryless* policies, which map an observation to a decision. In particular, the *stationary* policies, i.e., $\delta^t = \delta^1$ for all t in $[T]$, are much more appreciated in the reinforcement learning community [108].

Weakly coupled dynamic programs. As mentioned in Example 1, the predictive maintenance problem with capacity constraints can also be formalized as a restless partially observable multi-armed bandits problem, which is a special case of the weakly coupled POMDP problem.

When the decision maker observes the arm states, the problem corresponds to a restless multi-armed bandit. Such a problem lies in the broad class of multi-stage stochastic optimization problems that decompose into independent subproblems, which are linked by a collection of constraints on the action space. Precisely, each subsystem is a MDP on disjoint state spaces but the actions taken on each subsystem at each time are linked by a collection of constraints. When these constraints bind weakly, such a problem becomes a weakly coupled dynamic program [2, 105]. It enables to model various types of practical problems such as stochastic inventory routing with limited vehicle capacity [74], stochastic multi-product dispatch problems [112], scheduling problems [162], resources allocation [49], revenue management [152] and others. More examples can be found for instance in the dissertation of Hawkins [55].

In our case, the decision maker has only access to a partial observation of the arm's state. If the arms are rested, such a problem has been introduced as POMDP multi-armed bandit problem by Krishnamurthy and Wahlberg [79]. If the arms are restless, like in our maintenance problem, our formalization of the predictive maintenance problem is related to the model considered in the recent works of Mehta et al. [101], Meshram et al. [104], Mehta et al. [102] and Abbou and Makis [1]. In these works, the model considered has at most two states and three observations, which is lower than the number of states and observations we consider in practice (see Part III). Parizi and Ghate [116] recently proposed a similar framework for weakly coupled POMDP problem where the decision maker maximizes his expected total discounted reward over infinite horizon.

4 Integer programming for POMDPs

We have seen in Chapter 3 that the POMDP problem with memoryless policies (P_{ml}), which we recall below, is suitable to formalize the predictive maintenance problem with capacity constraints.

$$\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right] \tag{P_{ml}}$$

Since (P_{ml}) is NP-hard [87], it leaves the question of how to solve it. To the best of our knowledge, there is no integer programming approach in the literature addressing the POMDP problem with memoryless policies. In this chapter, we introduce an mixed-integer linear program (MILP) that models (P_{ml}). In addition, we introduce some valid inequalities improving the relaxation of the MILP. These valid inequalities come from a probabilistic interpretation of the dependence between the random variables in the POMDP. In Part II, we will show that we can extend these valid inequalities to any stochastic optimization problem where we have access to the probabilistic dependences between the random variables of the problem. The numerical experiments show that the MILP together with the valid inequalities can be solved using off-the-shell solvers. We also show in this chapter that the linear relaxation of our integer formulation corresponds to the so-called *MDP approximation* [54], i.e., the fully observed MDP relaxation of the POMDP. Then, we use this result to give guarantees about the cost of using a memoryless policy instead of a history-dependent policy. Chapter 4 is organized as follows.

- Section 4.1 introduces an MILP that exactly models (P_{ml}).
- Section 4.2 introduces an extended formulation with new valid inequalities that improve the resolution of the MILP.
- Section 4.3 shows how our MILP is related to POMDP with history-dependent policies ($\mathrm{P}_{\mathrm{his}}$). In particular it gives an interpretation in terms of information relaxation.
- Section 4.4 introduces an MILP that also models (P_{ml}).
- Section 4.5 presents numerical results, showing the efficiency of the approach.

4.1 Integer program for POMDPs with memoryless policies

We are given a POMDP $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$ and a finite horizon $T \in \mathbb{Z}_+$. We denote by v^*_{ml} the optimal value of (P_{ml}).

4.1.1 An exact Nonlinear Program (NLP)

We introduce the following NLP with a collection of variables
$\boldsymbol{\mu} = \big((\mu_s^1)_s, \big((\mu_{soa}^t)_{s,o,a}, (\mu_{sas'}^t)_{s,a,s'}\big)_t\big)$, $\boldsymbol{\delta} = \Big(\Big(\delta_{a|o}^t\Big)_{a,o}\Big)_t$.

$$\max_{\boldsymbol{\mu},\boldsymbol{\delta}} \sum_{t=1}^{T} \sum_{\substack{s,s' \in \mathcal{X}_S \\ a \in \mathcal{X}_A}} r(s,a,s')\mu_{sas'}^t \tag{4.1a}$$

$$\text{s.t.} \sum_{o \in \mathcal{X}_O} \mu_{soa}^t = \sum_{s' \in \mathcal{X}_S} \mu_{sas'}^t \qquad \forall s \in \mathcal{X}_S, a \in \mathcal{X}_A, t \in [T] \tag{4.1b}$$

$$\sum_{s \in \mathcal{X}_S, a \in \mathcal{X}_A} \mu_{sas'}^t = \sum_{o \in \mathcal{X}_O, a \in \mathcal{X}_A} \mu_{s'oa}^{t+1} \qquad \forall s' \in \mathcal{X}_S, t \in [T] \tag{4.1c}$$

$$\mu_s^1 = \sum_{o \in \mathcal{X}_O, a \in \mathcal{X}_A} \mu_{soa}^1 \qquad \forall s \in \mathcal{X}_S \tag{4.1d}$$

$$\mu_s^1 = p(s) \qquad \forall s \in \mathcal{X}_S \tag{4.1e}$$

$$\mu_{sas'}^t = p(s'|s,a) \sum_{s'' \in \mathcal{X}_S} \mu_{sas''}^t \qquad \forall s,s' \in \mathcal{X}_S, a \in \mathcal{X}_A, t \in [T] \tag{4.1f}$$

$$\mu_{soa}^t = \delta_{a|o}^t p(o|s) \sum_{o' \in \mathcal{X}_O, a' \in \mathcal{X}_A} \mu_{so'a'}^t \quad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{4.1g}$$

$$\boldsymbol{\delta} \in \Delta_{\mathrm{ml}} \tag{4.1h}$$

Given a policy $\boldsymbol{\delta} \in \Delta_{\mathrm{ml}}$, we say that $\boldsymbol{\mu}$ is the vector of *moments* of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$ when

$$\begin{aligned}
\mu_s^1 &= \mathbb{P}_{\boldsymbol{\delta}}(S_1 = s), && \forall s \in \mathcal{X}_S \\
\mu_{soa}^t &= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a), && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, \forall t \in [T] \\
\mu_{sas'}^t &= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s'), && \forall s,s' \in \mathcal{X}_S, a \in \mathcal{X}_A, \forall t \in [T].
\end{aligned} \tag{4.2}$$

Thanks to the properties of probability distributions, such vector of moments (4.2) of $\mathbb{P}_{\boldsymbol{\delta}}$ satisfy the constraints of NLP (4.1). Conversely, given a feasible solution of NLP (4.1), Theorem 4.1 ensures that $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$. This theorem tells even more. We denote by z^* the optimal value of NLP (4.1).

Theorem 4.1. *Let* $(\boldsymbol{\mu},\boldsymbol{\delta})$ *be a feasible solution of NLP* (4.1). *Then* $\boldsymbol{\mu}$ *is the vector of moments of the probability distribution* $\mathbb{P}_{\boldsymbol{\delta}}$ *induced by* $\boldsymbol{\delta}$*, and* $(\boldsymbol{\mu},\boldsymbol{\delta})$ *is an optimal solution of NLP* (4.1) *if and only if* $\boldsymbol{\delta}$ *is an optimal policy of* (P_{ml}). *In particular,* $v_{\mathrm{ml}}^* = z^*$.

Proof. Let $(\boldsymbol{\mu},\boldsymbol{\delta})$ be a feasible solution of NLP (4.1). We prove by induction on t that $\mu_s^1 = \mathbb{P}_{\boldsymbol{\delta}}\big(S_1 = s\big)$, $\mu_{soa}^t = \mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s, O_t = o, A_t = a\big)$ and $\mu_{sas'}^t = \mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s, A_t = a, S_{t+1} = s'\big)$. At time $t = 1$, the statement is immediate. Suppose that it holds up to $t-1$. Constraints (4.1g), (4.1c) and induction hypothesis ensure that

$$\begin{aligned}
\mu_{soa}^t = \delta_{a|o}^t p(o|s) \sum_{o',a'} \mu_{so'a'}^t = \delta_{a|o}^t p(o|s) \sum_{s',a'} \mu_{s'a's}^{t-1} &= \delta_{a|o}^t p(o|s) \sum_{s',a'} \mathbb{P}_{\boldsymbol{\delta}}\big(S_{t-1} = s', A_{t-1} = a', S_t = s\big) \\
&= \delta_{a|o}^t p(o|s) \mathbb{P}_{\boldsymbol{\delta}}(S_t = s) \\
&= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a),
\end{aligned}$$

where the last equality comes from the conditional independences and the law of total probability. Constraints (4.1b),(4.1f) and the induction hypothesis ensure that :

$$\mu^t_{sas'} = p(s'|s,a)\sum_{\overline{s}} \mu^t_{sa\overline{s}} = p(s'|s,a)\sum_{o} \mu^t_{soa} = p(s'|s,a)\sum_{o} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a)$$
$$= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s')$$

where the last equality comes from the conditional independences and the law of total probability. Consequently,

$$\sum_{t=1}^{T} \sum_{\substack{s,s'\in\mathcal{X}_S \\ a\in\mathcal{X}_A}} r(s,a,s')\mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s, A_t = a, S_{t+1} = s'\big) = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right],$$

which implies that $\boldsymbol{\delta}$ is optimal if and only if $(\boldsymbol{\mu},\boldsymbol{\delta})$ is optimal for NLP (4.1) and $v^*_{\mathrm{ml}} = z^*$. It achieves the proof. □

Remark 4. Suppose that the decision maker has access to an initial observation $\overline{o}$ in $\mathcal{X}_O$. Hence, for any policy $\boldsymbol{\delta}$ in Δ_{ml} we have $\mathbb{P}_{\boldsymbol{\delta}}(O_1 = \overline{o}) = 1$. Taking into account the initial observation requires to slightly modify the constraints of NLP (4.1): We replace constraints (4.1e) and (4.1g) in NLP (4.1) at time $t = 1$ by

$$\begin{aligned} &\mu^1_s = p(s|\overline{o}), && \forall s \in \mathcal{X}_S, \\ &\mu^1_{soa} = \delta^1_{a|o}\mathbb{1}_{\overline{o}}(o) \sum_{o'\in\mathcal{X}_O, a'\in\mathcal{X}_A} \mu^1_{so'a'}, && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A. \end{aligned} \tag{4.3}$$

This remark will be useful in Chapter 5. △

4.1.2 Turning the NLP into an MILP

We define the set of deterministic memoryless policies $\Delta^{\mathrm{d}}_{\mathrm{ml}}$ as

$$\Delta^{\mathrm{d}}_{\mathrm{ml}} = \left\{\boldsymbol{\delta} \in \Delta_{\mathrm{ml}} \colon \delta^t_{a|o} \in \{0,1\},\ \forall o \in \mathcal{X}_O,\ \forall a \in \mathcal{X}_A,\ \forall t \in [T]\right\}. \tag{4.4}$$

The following proposition states that we can restrict our policy search in (P_{ml}) to the set of deterministic memoryless policies.

Proposition 4.2. *[9, Proposition 1] There always exists an optimal policy for* (P_{ml}) *that is deterministic, i.e.,*

$$\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right] = \max_{\boldsymbol{\delta}\in\Delta^{\mathrm{d}}_{\mathrm{ml}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right]. \tag{4.5}$$

Theorem 4.1 ensures that (P_{ml}) and NLP (4.1) are equivalent, and in particular admit the same optimal solution in terms of $\boldsymbol{\delta}$. However, NLP (4.1) is hard to solve due to the nonlinear constraints (4.1g). By Proposition 4.5, we can add the integrality constraints of $\Delta^{\mathrm{d}}_{\mathrm{ml}}$ in (4.1), and, by a classical result in integer programming, we can turn NLP (4.1) into an equivalent MILP by replacing constraint (4.1g) by its McCormick relaxation [100], without changing its optimal value. The McCormick's inequalities for Equation (4.1g) are

$$\mu^t_{soa} \leqslant p(o|s) \sum_{o' \in \mathcal{X}_O, a' \in \mathcal{X}_A} \mu^t_{so'a'} \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{4.6a}$$

$$\mu^t_{soa} \leqslant \delta^t_{a|o} \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{4.6b}$$

$$\mu^t_{soa} \geqslant p(o|s) \sum_{o' \in \mathcal{X}_O, a' \in \mathcal{X}_A} \mu^t_{so'a'} + \delta^t_{a|o} - 1 \quad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T]. \tag{4.6c}$$

For convenience, we denote by McCormick$\big(\boldsymbol{\mu},\boldsymbol{\delta}\big)$ the set of McCormick linear inequalities (4.6). Thus, by using McCormick's linearization on constraints (4.1g), we get that (P$_{\mathrm{ml}}$) is equivalent to the following MILP:

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\delta}} \ & \sum_{t=1}^{T} \sum_{\substack{s,s' \in \mathcal{X}_S \\ a \in \mathcal{X}_A}} r(s,a,s') \mu^t_{sas'} \\ \text{s.t.} \ & \boldsymbol{\mu} \text{ satisfies (4.1b) – (4.1f)} \\ & \text{McCormick}\big(\boldsymbol{\mu},\boldsymbol{\delta}\big) \\ & \boldsymbol{\delta} \in \Delta^{\mathrm{d}}_{\mathrm{ml}} \\ & \boldsymbol{\mu} \geqslant 0. \end{aligned} \tag{4.7}$$

4.2 Valid cuts

To introduce our valid cuts in this section, we start by explaining why the linear relaxation of MILP (4.7) is not sufficient to define a feasible solution of NLP (4.1).

It turns out that given a feasible solution $(\boldsymbol{\mu},\boldsymbol{\delta})$ of the linear relaxation of MILP (4.7), the vector $\boldsymbol{\mu}$ is not necessarily the vector of moments of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$. Indeed, since the coordinates of the vector $\boldsymbol{\delta}$ are continuous variables, the McCormick's constraints (4.6) are, in general, no longer equivalent to the bilinear constraints (4.1g). Then, $(\boldsymbol{\mu},\boldsymbol{\delta})$ is no longer a feasible solution of NLP (4.1), which implies that $\boldsymbol{\mu}$ is not the vector of moments of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$.

Intuitively, it means that the feasible set of the linear relaxation of MILP (4.7) is too large. Actually, we can reduce the feasible set of the linear relaxation of MILP (4.7) by adding valid cuts. To do so, we introduce new variables $\big((\mu^t_{s'a'soa})_{s',a',s,o,a}\big)_t$ and the equalities

$$\sum_{s' \in \mathcal{X}_S, a' \in \mathcal{X}_A} \mu^t_{s'a'soa} = \mu^t_{soa}, \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, \tag{4.8a}$$

$$\sum_{a \in \mathcal{X}_A} \mu^t_{s'a'soa} = p(o|s)\mu^{t-1}_{s'a's}, \qquad \forall s', s \in \mathcal{X}_S, o \in \mathcal{X}_O, a' \in \mathcal{X}_A, \tag{4.8b}$$

$$\mu^t_{s'a'soa} = p(s|s',a',o) \sum_{\overline{s} \in \mathcal{X}_S} \mu^t_{s'a'\overline{s}oa}, \forall s', s \in \mathcal{X}_S, o \in \mathcal{X}_O, a', a \in \mathcal{X}_A, \tag{4.8c}$$

where

$$p(s|s',a',o) = \mathbb{P}(S_t = s | S_{t-1} = s', A_{t-1} = a', O_t = o),$$

for any $s, s' \in \mathcal{X}_S$, $a' \in \mathcal{X}_A$ and $o \in \mathcal{X}_O$. Note that $p(s|s',a',o')$ does not depend on the policy $\boldsymbol{\delta}$ and can be easily computed during a preprocessing using Bayes rules. Therefore, constraints in (4.8) are linear.

Proposition 4.3. *Equalities* (4.8) *are valid for MILP* (4.7)*, and there exists solution* $\boldsymbol{\mu}$ *of the linear relaxation of* (4.7) *that does not satisfy constraints* (4.8)*.*

The MILP formulation obtained by adding equalities (4.8) in MILP (4.7) is an extended formulation, and has much more constraints than the initial MILP (4.7). Its linear relaxation therefore takes longer to solve. Equalities (4.8) strengthen the linear relaxation, and numerical experiments in Section 4.5 show that these equalities enable to speed up the resolution of MILP (4.7).

Proof of Proposition 4.3. Let $(\boldsymbol{\mu}, \boldsymbol{\delta})$ be a feasible solution of MILP (4.7). We define

$$\mu^t_{s'a'soa} = \delta^t_{a|o} p(o|s) \mu^{t-1}_{s'a's}$$

for all $(s', a', s, o, a) \in \mathcal{X}_S \times \mathcal{X}_A \times \mathcal{X}_S \times \mathcal{X}_O \times \mathcal{X}_A$, $t \in [T]$. These new variables satisfy constraints in (4.8) :

$$\sum_{a \in \mathcal{X}_A} \mu^t_{s'a'soa} = \left(\sum_{a \in \mathcal{X}_A} \delta^t_{a|o} \right) p(o|s) \mu^{t-1}_{s'a's} = p(o|s) \mu^{t-1}_{s'a's}$$

$$\sum_{a' \in \mathcal{X}_A, s' \in \mathcal{X}_S} \mu^t_{s'a'soa} = \left(\sum_{a' \in \mathcal{X}_A, s' \in \mathcal{X}_S} \mu^{t-1}_{s'a's} \right) \delta^t_{a|o} p(o|s) = \delta^t_{a|o} p(o|s) \sum_{o' \in \mathcal{X}_O, a' \in \mathcal{X}_O} \mu^t_{so'a'} = \mu^t_{soa}$$

The remaining constraint (4.8c) is obtained using the following observation :

$$\frac{\mu^t_{s'a'soa}}{\sum_{s'' \in \mathcal{X}_S} \mu^t_{s'a's''oa}} = \frac{p(o|s) \mu^{t-1}_{s'a's}}{\sum_{s'' \in \mathcal{X}_S} p(o|s'') \mu^{t-1}_{s'a's''}} = \frac{p(o|s) p(s|s', a') \sum_{\overline{s}} \mu^{t-1}_{s'a'\overline{s}}}{\sum_{s'' \in \mathcal{X}_S} p(o|s'') p(s''|s', a') \sum_{\overline{s}} \mu^{t-1}_{s'a'\overline{s}}} = \frac{p(o|s) p(s|s', a')}{\sum\limits_{s'' \in \mathcal{X}_S} p(o|s'') p(s''|s', a')}$$

By setting $p(s|s', a', o) = \dfrac{p(o|s) p(s|s', a')}{\sum\limits_{\overline{s} \in \mathcal{X}_S} p(o|\overline{s}) p(\overline{s}|s', a')}$, equality (4.8c) holds.

Now we prove that there exists a solution $\boldsymbol{\mu}$ of the linear relaxation of MILP (4.7) that does not satisfy equalities (4.8). We define such a solution $(\boldsymbol{\mu}, \boldsymbol{\delta})$. We set $\mu^1_s = p(s)$ for all s in $\mathcal{X}_S$, and for all t in $[T]$:

$$\mu^1_{soa} = \begin{cases} p(o|s) \mu^1_s, \text{ if } a = \phi(s) \\ 0, \text{ otherwise} \end{cases}, \qquad \text{if } t = 1,$$

$$\mu^t_{sas'} = p(s'|s, a) \sum_{o \in \mathcal{X}_O} \mu^t_{soa}$$

$$\mu^t_{soa} = \begin{cases} p(o|s) \sum_{s' \in \mathcal{X}_S, a' \in \mathcal{X}_A} \mu^{t-1}_{s'a's}, \text{ if } a = \phi(s) \\ 0, \text{ otherwise} \end{cases}, \qquad \text{if } t \geqslant 2, \tag{4.9}$$

$$\delta^t_{a|o} = \begin{cases} \frac{\sum_{s \in \mathcal{X}_S} \mu^t_{soa}}{\sum_{s \in \mathcal{X}_S, a \in \mathcal{X}_A} \mu^t_{soa}} \text{ if } \sum_{s \in \mathcal{X}_S, a \in \mathcal{X}_A} \mu^t_{soa} \neq 0 \\ \mathbb{1}_{\tilde{a}}(a), \text{ otherwise} \end{cases} \tag{4.10}$$

where $\phi : \mathcal{X}_S \to \mathcal{X}_A$ is an arbitrary mapping and $\tilde{a}$ is an arbitrary element in $\mathcal{X}_A$. We prove that

$\boldsymbol{\mu}$ is a feasible solution of the linear relaxation of MILP (4.7).

First, it is easy to see constraints (4.1e)-(4.1f) are satisfied. It remains to prove that constraints (4.6a), (4.6b), (4.6c) are satisfied. First, (4.6a) holds because

$$\mu^t_{soa} \leqslant \max\left(0, p(o|s) \sum_{s'\in\mathcal{X}_S, a'\in\mathcal{X}_A} \mu^{t-1}_{s'a's}\right) \leqslant p(o|s) \sum_{s'\in\mathcal{X}_S, a'\in\mathcal{X}_A} \mu^{t-1}_{s'a's},$$

Second, (4.6b) holds because

$$\mu^t_{soa} \leqslant \sum_{s'\in\mathcal{X}_S} \mu^t_{s'oa} = \delta^t_{a|o} \sum_{s'\in\mathcal{X}_S} p(o|s') \sum_{s''\in\mathcal{X}_S, a''\in\mathcal{X}_A} \mu^{t-1}_{s''a''s'} \leqslant \delta^t_{a|o},$$

where we used definition (4.10) from the first to second line. Third, (4.6c) holds because

$$\begin{aligned}\mu^t_{soa} - p(o|s) \sum_{\substack{s'\in\mathcal{X}_S\\a'\in\mathcal{X}_A}} \mu^{t-1}_{s'a's} &\geqslant \sum_{s''\in\mathcal{X}_S} \overbrace{\mu^t_{s''oa} - p(o|s'') \sum_{s'\in\mathcal{X}_S, a'\in\mathcal{X}_A} \mu^{t-1}_{s'a's''}}^{\leqslant 0} \\ &= \sum_{s'', s'\in\mathcal{X}_S, a'\in\mathcal{X}_A} p(o|s'')\mu^{t-1}_{s'a's''}(\delta^t_{a|o} - 1) \\ &\geqslant \delta^t_{a|o} - 1,\end{aligned}$$

which yields (4.6c). Therefore, $(\boldsymbol{\mu}, \boldsymbol{\delta})$ is a solution of the linear relaxation of MILP (4.7).

Now, we prove that such a solution does not satisfy cuts (4.8). We define the new variables:

$$\mu^t_{s'a'soa} = \begin{cases} \mu^{t-1}_{s'a's} \frac{\mu^t_{soa}}{\sum_{o'\in\mathcal{X}_O, a'\in\mathcal{X}_A} \mu^t_{so'a'}} & \text{if } \sum_{o'\in\mathcal{X}_O, a'\in\mathcal{X}_A} \mu^t_{so'a'} \neq 0 \\ 0 \text{ otherwise} \end{cases}$$

Hence, $\boldsymbol{\mu}$ satisfies constraints (4.8a) and (4.8b). However, constraint (4.8c) is not satisfied in general. Indeed, since the mapping ϕ is arbitrary, we can set ϕ such that $p(s|s', a', o) > 0$ and $\mu^t_{s'a'soa} = 0$. Therefore, there exists a solution $\boldsymbol{\mu}$ of the linear relaxation of MILP (4.7) that does not satisfy cuts (4.8). It achieves the proof. □

Probabilistic interpretation. Given a feasible solution $(\boldsymbol{\mu}, \boldsymbol{\delta})$ of the linear relaxation of (4.7), $\boldsymbol{\mu}$ can be interpreted as the vector of moments of a probability distribution $\mathbb{Q}_{\boldsymbol{\mu}}$ over $(\mathcal{X}_S \times \mathcal{X}_O \times \mathcal{X}_A)^T \times \mathcal{X}_S$. However, as it has been mentioned above, the vector $\boldsymbol{\mu}$ does not necessarily correspond to the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$, which is due to the fact that $(\boldsymbol{\mu}, \boldsymbol{\delta})$ does not necessarily satisfy the nonlinear constraints (4.1g). Besides, constraints (4.1g) happen to be equivalent to the property that,

according to $\mathbb{Q}_{\boldsymbol{\mu}}$, action A_t is independent from state S_t given observation O_t. (4.11)

Hence, given a feasible solution $(\boldsymbol{\mu}, \boldsymbol{\delta})$ of the linear relaxation of MILP (4.7), the distribution $Q_{\boldsymbol{\mu}}$ does not necessarily satisfy the conditional independences (4.11). Remark that (4.11) implies the weaker result that,

$$\text{according to } \mathbb{Q}_{\boldsymbol{\mu}},\ A_t \text{ is independent from } S_t \text{ given } O_t,\ A_{t-1} \text{ and } S_{t-1}. \tag{4.12}$$

Proposition 4.3 says that the independences in (4.12) are not satisfied in general by a feasible solution $(\boldsymbol{\mu},\boldsymbol{\delta})$ of the linear relaxation of MILP (4.7), but that we can enforce them using linear equalities (4.8) on $(\boldsymbol{\mu},\boldsymbol{\delta})$ in an extended formulation.

4.3 Strengths of the relaxations

It turns out that the linear relaxation of MILP (4.7) is related to the fully observed POMDP, which corresponds to the case when the decision maker directly observes the state of the system at each time. Hence, the problem becomes a MDP and it is called the MDP approximation [54]. We introduce a collection of variables $\boldsymbol{\mu} = \big((\mu_s^1)_s, (\mu_{sas'}^t)_{s,a,s'})_t\big)$ and the following linear program which is known to solve exactly a MDP (e.g. d'Epenoux [38]).

$$\max_{\boldsymbol{\mu}} \sum_{t=1}^{T} \sum_{\substack{s,s'\in\mathcal{X}_S\\ a\in\mathcal{X}_A}} r(s,a,s')\mu_{sas'}^t \tag{4.13a}$$

$$\text{s.t. } \mu_s^1 = \sum_{a'\in\mathcal{X}_A, s'\in\mathcal{X}_S} \mu_{sa's'}^1 \qquad \forall s\in\mathcal{X}_S \tag{4.13b}$$

$$\sum_{s'\in\mathcal{X}_S, a'\in\mathcal{X}_A} \mu_{s'a's}^t = \sum_{a'\in\mathcal{X}_A, s'\in\mathcal{X}_S} \mu_{sa's'}^{t+1} \qquad \forall s\in\mathcal{X}_S, t\in[T] \tag{4.13c}$$

$$\mu_s^1 = p(s) \qquad \forall s\in\mathcal{X}_S \tag{4.13d}$$

$$\mu_{sas'}^t = p(s'|s,a)\sum_{s''\in\mathcal{X}_S} \mu_{sas''}^t \qquad \forall s\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T] \tag{4.13e}$$

In Linear program (4.13), the variables $(\mu_s^1)_s$ and $(\mu_{sas'}^t)_{sas'}$ respectively represent the probability distribution of S_1 and (S_t, A_t, S_{t+1}) for any t in $[T]$. Theorem 4.4 below states that the linear relaxation of MILP (4.7) is equivalent to the MDP approximation of $(\mathrm{P}_{\mathrm{ml}})$. We introduce the following quantities:

- z_{R}^*: the optimal value of the linear relaxation of MILP (4.7).
- $z_{\mathrm{R^c}}^*$: the optimal value of the linear relaxation of MILP (4.7) with inequalities (4.8).
- v_{his}^*: the optimal value of $(\mathrm{P}_{\mathrm{his}})$.
- v_{MDP}^*: the optimal value of linear program (4.13), which is the optimal value of the MDP approximation.

Theorem 4.4. *Let $(\boldsymbol{\mu},\boldsymbol{\delta})$ be feasible solution of the linear relaxation of MILP* (4.7)*. Then $(\boldsymbol{\mu},\boldsymbol{\delta})$ is an optimal solution of the linear relaxation of MILP* (4.7) *if and only if $\boldsymbol{\mu}$ is an optimal solution of linear program* (4.13)*. In particular, $z_{\mathrm{R}}^* = v_{\mathrm{MDP}}^*$. In addition, the following inequalities hold:*

$$z^* \leqslant v_{\mathrm{his}}^* \leqslant z_{\mathrm{R^c}}^* \leqslant z_{\mathrm{R}}^*. \tag{4.14}$$

Inequality (4.14) ensures that by solving MILP (4.7), we obtain an integrality gap $z_{\mathrm{R}}^* - z^*$ that bounds the approximation error $v_{\mathrm{his}}^* - z^*$ due to the choice of a memoryless policy instead of a policy that depends on all history of observations and actions. In addition, Theorem 4.4

ensures that the integrality gap $z^*_{\mathrm{R}^c} - z^*$ obtained using valid inequalities (4.8) gives a tighter bound on the approximation error.

Proof of Theorem 4.4. We first prove the equivalence between the linear relaxation of our MILP (4.7) and its MDP approximation. Note that in both problem, the objective function are the same. Hence, we only need to prove that we can construct a feasible solution from a problem to another.

Let $(\boldsymbol{\mu}, \boldsymbol{\delta})$ be a feasible solution of the linear relaxation of MILP (4.7). Constraints (4.1b)- (4.1f) ensure that $(\mu^1_s, \mu^t_{sas'})_{t\in[T]}$ satisfies constraints (4.13d)-(4.13e), which implies that it is a feasible solution of Linear program (4.13).

Let $\boldsymbol{\mu}$ be a feasible solution of Linear program (4.13). It suffices to define variables $\delta^t_{a|o}$ and μ^t_{soa} for all a in $\mathcal{X}_A$, o in $\mathcal{X}_O$, s in $\mathcal{X}_S$, and t in $[T]$. We define these variables using (4.9) and (4.10). In the proof of Proposition 4.3, we proved that $(\boldsymbol{\mu}, \boldsymbol{\delta})$ is a feasible solution of the linear relaxation of MILP (4.7). Consequently, the equivalence holds and $z^*_{\mathrm{R}} = v^*_{\mathrm{MDP}}$.

Now we prove that inequalities (4.14) hold. Note that Proposition 4.3 ensures that

$$z^* \leqslant z^*_{\mathrm{R}^c} \leqslant z^*_{\mathrm{R}}.$$

It remains to prove the two following inequalities.

$$z^* \leqslant v^*_{\mathrm{his}} \tag{4.15}$$

$$v^*_{\mathrm{his}} \leqslant z^*_{\mathrm{R}^c} \tag{4.16}$$

First, we prove Inequality (4.15). By definition, we have $\Delta_{\mathrm{ml}} \subseteq \Delta_{\mathrm{his}}$. Hence, we obtain $v^*_{\mathrm{ml}} \leqslant v^*_{\mathrm{his}}$. Using Theorem 4.1, we deduce that $z^* \leqslant v^*_{\mathrm{his}}$. Therefore the inequality $v^*_{\mathrm{ml}} \leqslant v^*_{\mathrm{his}} \leqslant z^*_{\mathrm{R}}$ holds.

Now we prove Inequality (4.16). The proof is based on a probabilistic interpretation of the valid inequalities (4.8). It suffices to proves that for any policy $\boldsymbol{\delta}$ in Δ_{his}, the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ satisfies the weak conditional independences (4.12). Let $\boldsymbol{\delta} \in \Delta_{\mathrm{his}}$. The probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ over the random variables $(S_t, A_t, O_t)_{1\leqslant t\leqslant T}$ according to $\boldsymbol{\delta}$ is exactly

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\delta}}((S_t = s_t, O_t = o_t, A_t = a_t)_{1\leqslant t\leqslant T}) = \mathbb{P}_{\boldsymbol{\delta}}(S_1 = s_1) \prod_{t=1}^{T} \mathbb{P}_{\boldsymbol{\delta}}(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)\\ \mathbb{P}_{\boldsymbol{\delta}}(O_t = o_t | S_t = s_t)\delta^t_{a_t|h_t}\end{aligned} \tag{4.17}$$

where $h_t = \{O_1 = o_1, A_1 = a_1, O_2 = o_2, \ldots, O_t = o_t\}$ is the history of observations and actions. Note that the policy at time t is the conditional probability $\delta^t_{a_t|h_t} = \mathbb{P}_{\boldsymbol{\delta}}(A_t = a_t | H_t = h_t)$. We define:

$$\begin{aligned}
\mu^1_s &= \mathbb{P}_{\boldsymbol{\delta}}(S_1 = s)\\
\mu^t_{soa} &= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a)\\
\mu^t_{sas'} &= \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s')\\
\mu^t_{s'a'soa} &= \mathbb{P}_{\boldsymbol{\delta}}(S_{t-1} = s', A_{t-1} = a', S_t = s, O_t = o, A_t = a)
\end{aligned}$$

We define the policy $\tilde{\boldsymbol{\delta}}$ using (4.10). It is easy to see that constraints of (4.7) are satisfied. Furthermore, we have $\tilde{\boldsymbol{\delta}} \in \Delta_{\mathrm{ml}}$. It remains to prove that equalities (4.8) are satisfied. By definition of a probability distribution, we directly see that constraints (4.8a) are satisfied. We prove (4.8b) and (4.8c). We compute the left-hand side of (4.8b):

$$\begin{aligned}
&\sum_{a\in\mathcal{X}_A} \mu^t_{s'a'soa} = \sum_{a\in\mathcal{X}_A} \mathbb{P}_{\boldsymbol{\delta}}(S_{t-1}=s', A_{t-1}=a', S_t=s, O_t=o, A_t=a)\\
&= \sum_{a\in\mathcal{X}_A} \sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}}\\
&\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-2}, S_{t-1}=s', O_{t-1}=o', A_{t-1}=a', S_t=s, O_t=o, A_t=a)\\
&= p(o|s)p(s|s',a') \sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-2}, S_{t-1}=s', O_{t-1}=o_{t-1}, A_{t-1}=a')\\
&\sum_{a\in\mathcal{X}_A} \delta_{a|h_t}\\
&= p(o|s)p(s|s',a') \sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-2}, S_{t-1}=s', O_{t-1}=o', A_{t-1}=a')\\
&= p(o|s)p(s|s',a')\mathbb{P}_{\boldsymbol{\delta}}(S_{t-1}=s', A_{t-1}=a')\\
&= p(o|s)\mu^{t-1}_{s'a's}
\end{aligned}$$

where we used the definition of the probability distribution (4.17) at the third equation. Therefore, constraints (4.8b) are satisfied by $\boldsymbol{\mu}$. To prove that constraints (4.8c) are satisfied, we prove that

$$\mathbb{P}_{\boldsymbol{\delta}}(S_t=s_t|S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, O_t=o_t, A_t=a_t) = \mathbb{P}_{\boldsymbol{\delta}}(S_t=s_t|S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, O_t=o_t)$$

We compute $\mathbb{P}_{\boldsymbol{\delta}}(S_t=s_t|S_{t-1}=s', A_{t-1}=a', O_t=o, A_t=a)$:

$$\begin{aligned}
&\mathbb{P}_{\boldsymbol{\delta}}(S_t=s_t|S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, O_t=o_t, A_t=a_t)\\
&= \frac{\mathbb{P}_{\boldsymbol{\delta}}(S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, S_t=s_t, O_t=o_t, A_t=a_t)}{\mathbb{P}_{\boldsymbol{\delta}}(S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, O_t=o_t, A_t=a_t)}\\
&= \frac{\sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t})}{\sum_{\substack{s_1,\dots,s_{t-2},s_t\\ h_{t-1}}} \mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t})}\\
&= \frac{\sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \delta^t_{a_t|h_t} p(o_t|s_t)p(s_t|s_{t-1},a_{t-1})\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-1})}{\sum_{\substack{s_1,\dots,s_{t-2},s'_t\\ h_{t-1}}} \delta^t_{a_t|h_t} p(o_t|s_t)p(s_t|s_{t-1},a_{t-1})\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-1})}\\
&= \frac{p(o_t|s_t)p(s_t|s_{t-1},a_{t-1})\sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \delta^t_{a_t|h_t}\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-1})}{\sum_{s'_t} p(o_t|s'_t)p(s'_t|s_{t-1},a_{t-1})\sum_{\substack{s_1,\dots,s_{t-2}\\ h_{t-1}}} \delta^t_{a_t|h_t}\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-1})}\\
&= \frac{p(o_t|s_t)p(s_t|s_{t-1},a_{t-1})}{\sum_{s'_t} p(o_t|s'_t)p(s'_t|s_{t-1},a_{t-1})}
\end{aligned}$$

where the last line goes from the fact that the term $\delta^t_{a_t|h_t}\mathbb{P}_{\boldsymbol{\delta}}((S_i=s_i,O_i=o_i,A_i=a_i)_{1\leqslant i\leqslant t-1})$ does not depend on s_t. Hence, constraints (4.8c) are satisfied by $\boldsymbol{\mu}$. We deduce that $\boldsymbol{\mu}$ is a feasible solution of MILP (4.7) satisfying the valid inequalities (4.8). Therefore,

$$\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right] = \sum_{t=1}^{T}\sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s')r(s,a,s') \leqslant z^*_{\mathrm{R^c}}$$

By maximizing over $\boldsymbol{\delta}$ the left-hand side, we obtain $v^*_{\mathrm{his}} \leqslant z^*_{\mathrm{R^c}}$. It achieves the proof. □

4.4 Value functions for POMDPs with memoryless policies

The aim of this section is to introduce an MILP that models ($\mathrm{P_{ml}}$) using a new type of variables. The formulation obtained is related to the linear programming duality and the link with the linear programming will be describe in Part II.

Like in Section 4.1, we introduce a NLP, then we get an MILP using McCormick inequalities and finally valid inequalities. In this section, some of the proofs will be written from a probabilistic point of view. We could have used this kind of proofs in Section 4.1.

4.4.1 An exact NLP

We introduce a collection of variables $\boldsymbol{\lambda} = \big((\lambda_s)_s, \big((\lambda^t_{soa})_{s,o,a}, (\lambda^t_{sas'})_{s,a,s'}\big)_t\big)$, $\boldsymbol{\delta} = \Big(\Big(\delta^t_{a|o}\Big)_{o,a}\Big)_t$ and the following NLP.

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \quad & \sum_{s\in\mathcal{X}_S} p(s)\lambda_s && && (4.18a)\\
\text{s.t.} \quad & \lambda_s = \sum_{o\in\mathcal{X}_O, a\in\mathcal{X}_A} p(o|s)\delta^1_{a|o}\lambda^1_{soa} && \forall s\in\mathcal{X}_S && (4.18b)\\
& \lambda^t_{soa} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\lambda^t_{sas'} && \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] && (4.18c)\\
& \lambda^t_{sas'} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O, a'\in\mathcal{X}_A} p(o'|s')\delta^{t+1}_{a'|o'}\lambda^{t+1}_{s'o'a'} && \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T-1] && (4.18d)\\
& \lambda^T_{sas'} = r(s,a,s') && \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A && (4.18e)\\
& \boldsymbol{\delta}\in\Delta_{\mathrm{ml}}. && && (4.18f)
\end{aligned}$$

Given $\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}$, we say that $\boldsymbol{\lambda}$ is the vector of *value functions* of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$ when

$$\lambda_s = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=1}^{T} r(S_{t'}, A_{t'}, S_{t'+1})|S_1 = s\right], \qquad \forall s\in\mathcal{X}_S \quad (4.19a)$$

$$\lambda^t_{soa} = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})|S_t = s, O_t = o, A_t = a\right], \quad \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] \quad (4.19b)$$

$$\lambda^t_{sas'} = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})|S_t = s, A_t = a, S_{t+1} = s'\right], \quad \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T]. \quad (4.19c)$$

Thanks to the properties of the conditional expectation, the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$ satisfies the constraints of NLP (4.18). Conversely, given a feasible solution $(\boldsymbol{\lambda},\boldsymbol{\delta})$ of NLP (4.18), Theorem 8.17 below ensures that $\boldsymbol{\lambda}$ is the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$. This theorem tells even more. We denote by z^*_{vf} the optimal value of NLP (4.18).

Theorem 4.5. *Let* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *be a feasible solution of NLP* (4.18)*. Then* $\boldsymbol{\lambda}$ *is the vector of value functions* (4.19) *of the distribution* $\mathbb{P}_{\boldsymbol{\delta}}$*, and* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *is an optimal solution of NLP* (4.18) *if and only if* $\boldsymbol{\delta}$ *is an optimal policy of* $(\mathrm{P}_{\mathrm{ml}})$*. In particular,* $v^*_{\mathrm{ml}} = z^*_{\mathrm{vf}}$.

Proof. Let $(\boldsymbol{\lambda},\boldsymbol{\delta})$ be a feasible solution of NLP (4.18). Then $\boldsymbol{\delta}$ is a feasible solution of $(\mathrm{P}_{\mathrm{ml}})$. We now prove that $\boldsymbol{\lambda}$ satisfies (4.19), from which we can deduce that $\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^T r(S_t,A_t,S_{t+1})\right] = \sum_{s\in\mathcal{X}_S} p(s)\lambda_s$. We prove the result using a backward induction from $t=T$ until $t=1$, where the induction hypothesis at time t is that $\boldsymbol{\lambda}$ satisfies (4.19) for t' in $[\![t,T]\!]$. At time $t=T$, the induction hypothesis is true.

We assume that the induction hypothesis is true until $t+1 \leqslant T$. At time t, constraints (4.18e) ensure that

$$\begin{aligned}
\lambda^t_{sas'} &= r(s,a,s') + \sum_{o'\in\mathcal{X}_O,a'\in\mathcal{X}_A} p(o'|s')\delta^{t+1}_{a'|o'}\lambda^{t+1}_{s'o'a'}\\
&= r(s,a,s') + \sum_{o'\in\mathcal{X}_O,a'\in\mathcal{X}_A} p(o'|s')\delta^{t+1}_{a'|o'}\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t+1} r(S_{t'},A_{t'},S_{t'+1})|S_{t+1}=s',O_{t+1}=o',A_{t+1}=a'\right]\\
&= r(s,a,s') + \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t+1} r(S_{t'},A_{t'},S_{t'+1})|S_{t+1}=s'\right]\\
&= r(s,a,s') + \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t+1} r(S_{t'},A_{t'},S_{t'+1})|S_t=s,A_t=a,S_{t+1}=s'\right]\\
&= \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t} r(S_{t'},A_{t'},S_{t'+1})|S_t=s,A_t=a,S_{t+1}=s'\right]
\end{aligned}$$

for all $s,s'\in\mathcal{X}_S$ and $a\in\mathcal{X}_A$. The fourth equality comes from the fact that all random variables $(S_{t'},O_{t'},A_{t'})_{t'\geqslant t+1}$ are conditionally independent from (S_t,A_t) given S_{t+1}. It proves that $\boldsymbol{\lambda}$ satisfies (4.19c). Constraints (4.18c) ensure that

$$\begin{aligned}
\lambda^t_{soa} &= \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\lambda^t_{sas'}\\
&= \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t} r(S_{t'},A_{t'},S_{t'+1})|S_t=s,A_t=a,S_{t+1}=s'\right]\\
&= \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t} r(S_{t'},A_{t'},S_{t'+1})|S_t=s,A_t=a\right] = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t} r(S_{t'},A_{t'},S_{t'+1})|S_t=s,O_t=o,A_t=a\right]
\end{aligned}$$

for all $s\in\mathcal{X}_S$, $o\in\mathcal{X}_O$, and $a\in\mathcal{X}_A$. It proves that $\boldsymbol{\lambda}$ satisfies (4.19b). In addition, constraints (4.18b) ensure that

$$\lambda_s = \sum_{o\in\mathcal{X}_O,a\in\mathcal{X}_A} p(o|s)\delta^1_{a|o}\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^T r(S_t,A_t,S_{t+1})|S_t=s,O_t=o\right] = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^T r(S_t,A_t,S_{t+1})|S_t=s\right].$$

It proves that $\boldsymbol{\lambda}$ satisfies (4.19a). Hence, $\boldsymbol{\lambda}$ is the vector of value functions of the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$.

Therefore, $\boldsymbol{\delta}$ is optimal if and only if $(\boldsymbol{\lambda},\boldsymbol{\delta})$ is optimal for NLP (4.18) and $v^*_{\mathrm{ml}} = z^*_{\mathrm{vf}}$. □

Remark 5. In fact, we can reduce the number of variables of NLP (4.18). Indeed, Theorem 8.17 ensures that if $(\boldsymbol{\lambda},\boldsymbol{\delta})$ is a feasible solution of NLP (4.18), then $\boldsymbol{\lambda}$ satisfies $\lambda^t_{soa} = \mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{t'=t}^{T} r(S_{t'},A_{t'},S_{t'+1}) | S_t = s, O_t = o, A_t = a\big]$ for every $s \in \mathcal{X}_S$, $o \in \mathcal{X}_O$, $a \in \mathcal{X}_A$ and $t \in [T]$. By definition of $\mathbb{P}_{\boldsymbol{\delta}}$, we have that the variables $(S_{t'},A_{t'},S_{t'+1})_{t'\geqslant t}$ are independent from O_t given (S_t,A_t). Therefore, the variables λ^t_{soa} do not depend on o, for any $s \in \mathcal{X}_S$, $a \in \mathcal{X}_A$ and $t \in [T]$. Consequently, we can replace the $T|\mathcal{X}_S||\mathcal{X}_O||\mathcal{X}_A|$ variables $\big((\lambda^t_{soa})_{s,o,a}\big)_t$, by the $T|\mathcal{X}_S||\mathcal{X}_A|$ new variables $\big((\lambda^t_{sa})_{s,a}\big)_t$. △

4.4.2 Turning the NLP into an MILP

In this section, we assume that we have access to a vector $\mathbf{b}^{\mathrm{lb}}$ and a vector $\mathbf{b}^{\mathrm{ub}}$ such that for every policy $\boldsymbol{\delta}$ in Δ_{ml},

$$\mathbf{b}^{\mathrm{lb}} \leqslant \boldsymbol{\lambda} \leqslant \mathbf{b}^{\mathrm{ub}},$$

where $\boldsymbol{\lambda}$ is the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$. In Section 4.4.3, we explain how we compute $\mathbf{b}^{\mathrm{lb}}$ and $\mathbf{b}^{\mathrm{ub}}$. These bounds play a key role because they drive the quality of the linear relaxation of MILP (4.21).

Theorem 8.17 ensures that NLP (4.18) exactly models $(\mathrm{P}_{\mathrm{ml}})$, and in particular both problems have the same optimal solution in terms of $\boldsymbol{\delta}$. However, NLP (4.18) is hard to solve due to the nonlinear constraints (4.18d). By Proposition 4.5, we can add the integrality constraints of $\Delta^{\mathrm{d}}_{\mathrm{ml}}$ in (4.18), and, by a classical result in integer programming, we can turn NLP (4.18) into an equivalent MILP by replacing the nonlinear terms in constraint (4.18d) by its McCormick relaxation, without changing its optimal value. To do so, we introduce variables $\boldsymbol{\alpha} = \big((\alpha^t_{soa})_{s,o,a}\big)_t$ and the following inequalities, which are the McCormick inequalities for constraints (4.18d).

$$\begin{aligned}
&\alpha^t_{soa} \leqslant b^{\mathrm{ub,t}}_{soa}\delta^t_{a|o} && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] && \text{(4.20a)}\\
&\alpha^t_{soa} \leqslant \lambda^t_{soa} && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] && \text{(4.20b)}\\
&\alpha^t_{soa} \geqslant \lambda^t_{soa} + b^{\mathrm{ub,t}}_{soa}(\delta^t_{a|o} - 1) && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] && \text{(4.20c)}\\
&\alpha^t_{soa} \geqslant b^{\mathrm{lb,t}}_{soa}\delta^t_{a|o} && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] && \text{(4.20d)}
\end{aligned}$$

For convenience, we denote by $\mathrm{McCormick}\big(\boldsymbol{\alpha},\boldsymbol{\lambda},\boldsymbol{\delta}\big)$ the McCormick linear inequalities (4.20). Thus, we get that $(\mathrm{P}_{\mathrm{ml}})$ is equivalent to the following MILP:

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\delta}} \quad & \sum_{s\in\mathcal{X}_S} p(s)\lambda_s \\
\text{s.t.} \quad & \lambda_s = \sum_{o\in\mathcal{X}_O, a\in\mathcal{X}_A} p(o|s)\alpha^1_{soa} && \forall s \in \mathcal{X}_S\\
& \lambda^t_{sas'} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O, a'\in\mathcal{X}_A} p(o'|s)\alpha^{t+1}_{so'a'} && \forall s \in \mathcal{X}_S, t \in [T-1]\\
& \lambda^t_{soa} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\lambda^t_{sas'} && \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T]\\
& \mathrm{McCormick}\big(\boldsymbol{\alpha},\boldsymbol{\lambda},\boldsymbol{\delta}\big)\\
& \boldsymbol{\delta} \in \Delta^{\mathrm{d}}_{\mathrm{ml}}
\end{aligned} \tag{4.21}$$

Remark 6. Let $\boldsymbol{\delta}$ be a feasible solution of ($\mathrm{P_{ml}}$). Then, Theorem 4.1 and Theorem 8.17 respectively ensure that there exists a unique vector of moments $\boldsymbol{\mu}$ and a unique vector of value functions $\boldsymbol{\lambda}$ of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$. Furthermore, $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are linked by the following relation.

$$\sum_{s,s'\in\mathcal{X}_S,a\in\mathcal{X}_A} \lambda^t_{sas'}\mu^t_{sas'} = \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{t'=t}^{T} r(S_{t'},A_{t'},S_{t'+1})\Big], \qquad \forall t\in[T].$$

In Part II, we give another interpretation of the vector of value functions. It turns out that in some specific cases like MDPs, the value functions represent the variables of some related dual linear programs. △

4.4.3 Computing bounds on the value functions

The aim of this section is to explain how we compute the vector of lower bounds $\mathbf{b}^{\mathrm{lb}}$ and the vector of upper bounds $\mathbf{b}^{\mathrm{ub}}$ of the vector of value functions $\boldsymbol{\lambda}$. We define $\mathbf{b}^{\mathrm{lb}}$ as follows.

$$\begin{cases} b^{\mathrm{lb,T}}_{sas'} = r(s,a,s'), & \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A \\ b^{\mathrm{lb,t}}_{sas'} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O} p(o'|s')\min_{a'\in\mathcal{X}_A} b^{\mathrm{lb,t+1}}_{s'o'a'}, & \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T-1] \\ b^{\mathrm{lb,t}}_{soa} = \sum_{s'} p(s'|s,a) b^{\mathrm{lb,t}}_{s,a,s'}, & \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] \\ b^{\mathrm{lb}}_{s} = \sum_{o\in\mathcal{X}_O} p(o|s)\min_{a'\in\mathcal{X}_A} b^{\mathrm{lb,1}}_{s,o,a}, & \forall s\in\mathcal{X}_S \end{cases} \tag{4.22}$$

And we define $\mathbf{b}^{\mathrm{ub}}$ as follows.

$$\begin{cases} b^{\mathrm{ub,T}}_{sas'} = r(s,a,s'), & \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A \\ b^{\mathrm{ub,t}}_{sas'} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O} p(o'|s')\max_{a'\in\mathcal{X}_A} b^{\mathrm{ub,t+1}}_{s'o'a'}, & \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T-1] \\ b^{\mathrm{ub,t}}_{soa} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a) b^{\mathrm{ub,t}}_{s,a,s'}, & \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] \\ b^{\mathrm{ub}}_{s} = \sum_{o\in\mathcal{X}_O} p(o|s)\max_{a'\in\mathcal{X}_A} b^{\mathrm{ub,1}}_{s,o,a}, & \forall s\in\mathcal{X}_S \end{cases} \tag{4.23}$$

Proposition 4.6. *Let $\boldsymbol{\lambda}$ be the vector of value functions of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by a policy $\boldsymbol{\delta}$. Then, $\boldsymbol{\lambda}$ satisfies $\mathbf{b}^{\mathrm{lb}} \leqslant \boldsymbol{\lambda} \leqslant \mathbf{b}^{\mathrm{ub}}$.*

Proof. We prove the result for the upper bounds. It suffices to replace the max operator by min operator for the lower bounds. Let $(\boldsymbol{\lambda},\boldsymbol{\delta})$ be a feasible solution of MILP (4.21). We prove the result by induction on t. The result is true at $t=T$. Suppose that the induction hypothesis holds until t. By definition of the vector of value functions, we have:

$$\begin{aligned} \lambda^t_{sas'} = r(s,a,s') + \sum_{o',a'} p(o'|s')\delta^t_{a'|o'}\lambda^{t+1}_{s'o'a'} &\leqslant r(s,a,s') + \sum_{o',a'} p(o'|s')\delta^t_{a'|o'} b^{\mathrm{ub,t+1}}_{s'o'a'} \\ &\leqslant r(s,a,s') + \sum_{o'} p(o'|s')\max_{a'\in\mathcal{X}_A} b^{\mathrm{ub,t+1}}_{s'o'a'} = b^{\mathrm{ub,t}}_{sas'}, \end{aligned}$$

where the first inequality comes from the induction hypothesis and the second inequality comes from the fact that the $\boldsymbol{\delta}$ is deterministic. In addition, we have:

$$\lambda^t_{soa} = \sum_{s'} p(s'|s,a)\lambda^t_{sas'} \leqslant \sum_{s'} p(s'|s,a) b^{\mathrm{ub,t}}_{sas'} = b^{\mathrm{ub,t}}_{soa}$$

It achieves the proof. □

4.4.4 Strengthening the linear relaxation

In this section, we introduce an MILP that uses the conditional probabilities $\mathbb{P}(S_t|S_{t-1}, A_{t-1}, O_t)$, which appear in the valid inequalities (4.8), and leads to a linear relaxation that is tighter in practice.

Like for MILP (4.7), when $\boldsymbol{\delta}$ is a vector of continuous variables, the McCormick's constraints (4.20) do not ensure that the constraints $\alpha^t_{soa} = \lambda^t_{soa}\delta^t_{a|o}$ are satisfied for any s in $\mathcal{X}_S$, o in $\mathcal{X}_O$, a in $\mathcal{X}_A$ and t in $[T]$. Hence, the constraints (4.18d) are not necessary satisfied in the linear relaxation of MILP (4.21). Intuitively, it means that the feasible set of the linear relaxation of MILP (4.21) is too large. Like for MILP (4.7) we wish to reduce the feasible set of the linear relaxation of MILP (4.21). However, unlike MILP (4.7) we are not able to derive valid inequalities for the variables of MILP (4.21). Instead of introducing valid inequalities for MILP (4.21), we introduce a formulation that uses the conditional independences (4.12) and that gives McCormick's bound on the variables $\boldsymbol{\lambda}$ that are tighter.

We introduce this formulation in three steps to obtain another formulation that uses these independences. First, we introduce another NLP that gives an optimal strategy of $(\mathrm{P_{ml}})$. Second, we turn this NLP into an MILP using the McCormick's inequalities. Third, we compute a vector of lower bounds $\mathbf{b}^{\mathrm{lb,c}}$ and upper bounds $\mathbf{b}^{\mathrm{ub,c}}$ of any feasible solution and we prove that these bounds are respectively greater than $\mathbf{b}^{\mathrm{lb}}$ and smaller than $\mathbf{b}^{\mathrm{ub}}$.

A nonlinear formulation. We introduce the variables (λ^t_{soa}) and the following NLP:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \sum_{s\in\mathcal{X}_S} p(s)\lambda_s \tag{4.24a}$$

$$\text{s.t. } \lambda_s = \sum_{o\in\mathcal{X}_O, a\in\mathcal{X}_A} p(o|s)\delta^1_{a|o}\lambda^1_{soa} \qquad \forall s\in\mathcal{X}_S \tag{4.24b}$$

$$\lambda^t_{soa} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\lambda^t_{sas'} \qquad \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] \tag{4.24c}$$

$$\lambda^t_{sas'} = r(s,a,s') + \sum_{\substack{s''\in\mathcal{X}_S, o'\in\mathcal{X}_O,\\ a'\in\mathcal{X}_A}} p(o'|s')p(s''|s,a,o')\delta^{t+1}_{a'|o'}\lambda^{t+1}_{s''o'a'} \qquad \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A, t\in[T-1] \tag{4.24d}$$

$$\lambda^T_{sas'} = r(s,a,s') \qquad \forall s,s'\in\mathcal{X}_S, a\in\mathcal{X}_A \tag{4.24e}$$

$$\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}. \tag{4.24f}$$

Note that there always exists a feasible solution of NLP (4.24). The constraints of NLP (4.24) are similar to the ones of NLP (4.18) except that Constraints (4.24d) differ from Constraints (4.18d).

Indeed, Constraints (4.24d) can be written explicitly:

$$\lambda^t_{sas'} = r(s,a,s') + \sum_{o' \in \mathcal{X}_O, a' \in \mathcal{X}_A} \delta^{t+1}_{a'|o'} \mathbb{P}_{\boldsymbol{\delta}}(O_{t+1} = o' | S_{t+1} = s') \sum_{s'' \in \mathcal{X}_S} \mathbb{P}_{\boldsymbol{\delta}}(S_{t+1} = s'' | S_t = s, A_t = a, O_{t+1} = o') \lambda^{t+1}_{s''o'a'}$$

We replaced $\lambda^{t+1}_{s'o'a'}$ by the expected value $\sum_{s'' \in \mathcal{X}_S} \mathbb{P}_{\boldsymbol{\delta}}(S_{t+1} = s'' | S_t = s, A_t = a, O_{t+1} = o') \lambda^{t+1}_{s''o'a'}$. It turns out a feasible solution of NLP (4.24) is not necessarily a vector of value functions. Fortunately, Proposition 4.7 below ensures that despite the loss of the value function property of Theorem 8.17, any feasible policy $\boldsymbol{\delta}$ gives the same objective value for ($\mathrm{P_{ml}}$) and NLP (4.18).

Proposition 4.7. *Let* $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ *be a feasible solution of NLP* (4.24). *Then* $\boldsymbol{\lambda}$ *satisfies*

$$\mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})\Big] = \sum_{\substack{s,s' \in \mathcal{X}_S \\ a \in \mathcal{X}_A}} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') \lambda^t_{sas'}$$

$$\mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})\Big] = \sum_{\substack{s \in \mathcal{X}_S, o \in \mathcal{X}_O \\ a \in \mathcal{X}_A}} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a) \lambda^t_{soa}.$$

In particular, $\sum_{s \in \mathcal{X}_S} p(s) \lambda^1_s = \mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\big]$.

Proof. Let $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ be a feasible solution of NLP (4.24). We prove the result by a backward induction on t. It is immediate for $t = T$. Suppose that the result holds until $t+1$. We have

$$\begin{aligned}
&\sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') \lambda^t_{sas'} = \sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') r(s,a,s') \\
&+ \sum_{\substack{s,s',s'' \in \mathcal{X}_S, o' \in \mathcal{X}_O, \\ a,a' \in \mathcal{X}_A}} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a) \underbrace{p(s'|s,a) p(o'|s') p(s''|s,a,o')}_{= p(s'|s,a,o') p(o'|s') p(s'|s,a)} \delta^{t+1}_{a'|o'} \lambda^{t+1}_{s''o'a'} \\
&= \sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') r(s,a,s') \\
&+ \sum_{s,s'',o',a,a'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a) p(s''|s,a) p(o'|s'') \delta^{t+1}_{a'|o'} \underbrace{\sum_{s'} p(s'|s,a,o')}_{=1} \lambda^{t+1}_{s''o'a'} \\
&= \sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') r(s,a,s') \\
&+ \sum_{\substack{s,s',o' \\ a,a'}} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s', O_{t+1} = o', A_{t+1} = a') \lambda^{t+1}_{s'o'a'} \\
&= \sum_{s,a,s'} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, A_t = a, S_{t+1} = s') r(s,a,s') + \underbrace{\sum_{s',o',a'} \mathbb{P}_{\boldsymbol{\delta}}(S_{t+1} = s', O_{t+1} = o', A_{t+1} = a') \lambda^{t+1}_{s'o'a'}}_{= \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t'=t+1}^{T} r(S_{t'}, A_{t'}, S_{t'+1})\right]} \\
&= \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})\Big]
\end{aligned}$$

It follows that $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{t'=t}^{T} r(S_{t'}, A_{t'}, S_{t'+1})\big] = \sum_{s,o,a} \mathbb{P}_{\boldsymbol{\delta}}(S_t = s, O_t = o, A_t = a) \lambda^t_{soa}$. Finally, we have the following computation

$$\sum_s \lambda_s^1 p(s) = \sum_{s,o,a} p(s)p(o|s)\delta_{a|o}^1 \lambda_{soa}^1 = \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{t=1}^T r(S_t, A_t, S_{t+1})\Big],$$

which achieves the proof. □

Now, we are able to write a theorem for NLP (4.24) that is similar to Theorem 8.17. We denote by $z_{\mathrm{vf^c}}^*$ the optimal value of NLP (4.24).

Theorem 4.8. *Let* $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ *be a feasible solution of NLP* (4.24)*. Then,* $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ *is an optimal solution of NLP* (4.24) *if and only if* $\boldsymbol{\delta}$ *is an optimal policy of* $(\mathrm{P_{ml}})$*. In particular,* $v_{\mathrm{ml}}^* = z_{\mathrm{vf^c}}^*$.

Proof. The proof is immediate from Proposition 4.7. □

Turning NLP (4.24) **into an MILP.** Again, we can linearize the constraints (4.24d) by introducing the variables $\boldsymbol{\alpha}$ and the McCormick's inequalities (4.20). We obtain the following MILP:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\delta}} \sum_{s\in\mathcal{X}_S} p(s)\lambda_s \tag{4.25a}$$

$$\text{s.t. } \lambda_s = \sum_{o\in\mathcal{X}_O, a\in\mathcal{X}_A} p(o|s)\alpha_{soa}^1 \qquad \forall s\in\mathcal{X}_S \tag{4.25b}$$

$$\lambda_{sas'}^t = r(s,a,s') + \sum_{\substack{s''\in\mathcal{X}_S, o'\in\mathcal{X}_O\\ a'\in\mathcal{X}_A}} p(o'|s)p(s''|s,a,o')\alpha_{s''o'a'}^{t+1} \qquad \forall s\in\mathcal{X}_S, t\in[T-1] \tag{4.25c}$$

$$\lambda_{soa}^t = \sum_{s'\in\mathcal{X}_S} p(s'|s,a)\lambda_{sas'}^t \qquad \forall s\in\mathcal{X}_S, o\in\mathcal{X}_O, a\in\mathcal{X}_A, t\in[T] \tag{4.25d}$$

$$\mathrm{McCormick}\big(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\delta}\big)$$

$$\boldsymbol{\delta} \in \Delta_{\mathrm{ml}}^{\mathrm{d}}$$

Computing bounds. We define $\mathbf{b}^{\mathrm{lb,c}}$ and $\mathbf{b}^{\mathrm{ub,c}}$ as follows: For any s, s' in $\mathcal{X}_S$, o in $\mathcal{X}_O$ and a in $\mathcal{X}_A$,

$$\begin{cases} b_{sas'}^{\mathrm{lb,c},T} = r(s,a,s'), & \\ b_{sas'}^{\mathrm{lb,c},t} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O} p(o'|s') \min_{a'\in\mathcal{X}_A} \sum_{s''\in\mathcal{X}_S} p(s''|s,a,o') b_{s''o'a'}^{\mathrm{lb,c},t+1}, & \forall t\in[T-1], \\ b_{soa}^{\mathrm{lb,c},t} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a) b_{sas'}^{\mathrm{lb,c},t}, & \forall t\in[T], \end{cases}$$

and,

$$\begin{cases} b_{sas'}^{\mathrm{ub,c},T} = r(s,a,s'), & \\ b_{sas'}^{\mathrm{ub,c},t} = r(s,a,s') + \sum_{o'\in\mathcal{X}_O} p(o'|s') \max_{a'\in\mathcal{X}_A} \sum_{s''\in\mathcal{X}_S} p(s''|s,a,o') b_{s''o'a'}^{\mathrm{ub,c},t+1} & \forall t\in[T-1], \\ b_{soa}^{\mathrm{ub,c},t} = \sum_{s'\in\mathcal{X}_S} p(s'|s,a) b_{sas'}^{\mathrm{ub,c},t} & \forall t\in[T]. \end{cases}$$

Proposition 4.9. *Let* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *be a feasible solution of NLP* (4.24)*. Then,* $\boldsymbol{\lambda}$ *satisfies* $\mathbf{b}^{\mathrm{lb,c}} \leqslant \boldsymbol{\lambda} \leqslant \mathbf{b}^{\mathrm{ub,c}}$*. In addition, the bounds obtained are tighter than* $\mathbf{b}^{\mathrm{lb}}$ *and* $\mathbf{b}^{\mathrm{ub}}$*, i.e.,* $\mathbf{b}^{\mathrm{lb}} \leqslant \mathbf{b}^{\mathrm{lb,c}}$ *and* $\mathbf{b}^{\mathrm{ub}} \geqslant \mathbf{b}^{\mathrm{ub,c}}$.

As we will show in the numerical experiments in Section 4.5, the optimal value of the linear relaxation of MILP (4.25) is always non greater than the optimal value of the linear relaxation of MILP (4.21).

Proof of Proposition 4.9. We prove the result for $\mathbf{b}^{\mathrm{ub,c}}$. It will hold for $\mathbf{b}^{\mathrm{lb,c}}$ by reversing the inequality symbol and replacing the max operator by the min operator. Let $(\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\delta})$ be a feasible solution of MILP (4.25). It suffices to prove the result for the vector $(\lambda_{soa}^t)_{s,o,a,t}$ because Proposition 4.6 ensures that the other coordinates satisfy the inequality. Again, we prove the result by induction on t. The result holds for $t = T$. Suppose that the induction hypothesis holds until t. It implies that for any $s \in \mathcal{X}_S$, $o \in \mathcal{X}_O$ and $a \in \mathcal{X}_A$:

$$\begin{aligned}
\lambda_{soa}^t &= \sum_{s'} p(s'|s,a)\big(r(s,a,s') + \sum_{o'} p(o'|s') \sum_{s'',a} p(s''|s,a,o')\delta_{a'|o'}^t \lambda_{s''o'a'}^{t+1}\big) \\
&\leqslant \sum_{s'} p(s'|s,a)\big(r(s,a,s') + \sum_{o'} p(o'|s') \sum_{a'} \delta_{a'|o'}^t \sum_{s''} p(s''|s,a,o') b_{s''o'a'}^{\mathrm{ub},t+1}\big) \\
&\leqslant \sum_{s'} p(s'|s,a)\big(r(s,a,s') + \sum_{o'} p(o'|s') \max_{a'} \sum_{s''} p(s''|s,a,o') b_{s''o'a'}^{\mathrm{ub},t+1}\big) \\
&= b_{soa}^{\mathrm{ub,c},t}
\end{aligned}$$

Now we prove that $b_{soa}^{\mathrm{ub,c},t} \leqslant b_{soa}^{\mathrm{ub},t}$. Again, we prove the result by backward induction. It is immediate for $t = T$. Suppose that the induction hypothesis holds until $t+1$. We prove the result for t. By definition we have:

$$\begin{aligned}
b_{soa}^{\mathrm{ub,c},t} &= \sum_{s'} p(s'|s,a) r(s,a,s') + \sum_{s',o'} p(s'|s,a) p(o'|s') \max_{a'} \sum_{s''} p(s''|s,a,o') b_{s''o'a'}^{\mathrm{ub,c},t+1} \\
&\leqslant \sum_{s'} p(s'|s,a) r(s,a,s') + \sum_{s',s'',o'} p(s'|s,a) p(o'|s') p(s''|s,a,o') \max_{a'} b_{s''o'a'}^{\mathrm{ub,c},t+1} \\
&= \sum_{s'} p(s'|s,a) r(s,a,s') + \sum_{s'',o'} p(s''|s,a) p(o'|s'') \underbrace{\sum_{s'} p(s'|s,a,o')}_{=1} \max_{a'} b_{s''o'a'}^{\mathrm{ub},t+1} \\
&= b_{soa}^{\mathrm{ub},t},
\end{aligned}$$

The first inequality comes from the induction hypothesis and by decomposing the maximum over the sum. The second equality come from the fact that $p(s''|s,a,o')p(s'|s,a)p(o'|s') = p(s'|s,a,o')p(s''|s,a)p(o'|s'')$. It achieves the proof. □

4.5 Numerical experiments

We now provide experiments showing the practical efficiency of our approaches to POMDPs. In Section 4.5.1, we evaluate on random instances the performances of MILP (4.7) and the efficiency of the valid inequalities (4.8) to help its resolution. In Section 4.5.2, we evaluate the performances of MILP (4.7) on instances from the literature on POMDPs. In particular, we show that memoryless policies perform well on instances from maintenance problem. The instances can be found here.[1] All linear programs have been implemented in Julia with JuMP interface [41] and solved using `Gurobi` 9.0 [52]. Experiments have been run on a server with 192Gb of RAM and 32 cores at 3.30GHz.

4.5.1 Random instances

The instances are generated by first choosing $\mathrm{k}_s = |\mathcal{X}_S| = |\mathcal{X}_O|$ and $\mathrm{k}_a = |\mathcal{X}_A|$. We then randomly generate the initial probability $\big(p(s)\big)_{s\in\mathcal{X}_S}$, the transition probability $\big(p(s'|s,a)\big)_{\substack{s,s'\in\mathcal{X}_S\\ a\in\mathcal{X}_A}}$, the emission probability $\big(p(o|s)\big)_{\substack{s\in\mathcal{X}_S\\ o\in\mathcal{X}_O}}$ and the immediate reward function $\big(r(s,a,s')\big)_{\substack{s,s'\in\mathcal{X}_S\\ a\in\mathcal{X}_A}}$. An instance is the tuple $(\mathcal{X}_S,\mathcal{X}_O,\mathcal{X}_A,\mathfrak{p},\mathbf{r})$. A way to measure the difficulty of solving an MILP (4.7) for POMDP $(\mathcal{X}_S,\mathcal{X}_O,\mathcal{X}_A,\mathfrak{p},\mathbf{r})$ with horizon T can be characterized by the size of the set of deterministic policies $|\Delta^{\mathrm{d}}_{\mathrm{ml}}| = |\mathcal{X}_A|^{T|\mathcal{X}_O|}$, which only depends on the size of the observation space $\mathcal{X}_O$ and the action space $\mathcal{X}_A$. We generate instances for different values of the pair $(\mathrm{k}_s,\mathrm{k}_a)$.

Numerical experiments on random POMDP. We solve MILP (4.7) with and without valid equalities (4.8), MILP (4.7) (basic formulation) and MILP (4.25) (strengthened formulation). Algorithms were stopped after 3600s. Table 4.1 reports the results on the random instances. The first four columns indicate the size of state space $|\mathcal{X}_S|$, observation space $|\mathcal{X}_O|$, action space $|\mathcal{X}_A|$ and time horizon T. The fifth column indicates the mathematical program used to solve $(\mathrm{P}_{\mathrm{ml}})$, ether basic or strengthened. In the last three columns, we report the integrality gap, the final gap, the percentage of instances solved and the computation time. Note that for each instance of POMDP $(\mathcal{X}_S,\mathcal{X}_O,\mathcal{X}_A,\mathfrak{p},\mathbf{r})$, we solve it with horizon $T\in\{10,20\}$. For each tuple $(\mathcal{X}_S,\mathcal{X}_O,\mathcal{X}_A)$, the results in Table 4.1 are averaged over 30 instances $(\mathcal{X}_S,\mathcal{X}_O,\mathcal{X}_A,\mathfrak{p},\mathbf{r})$ where the $\mathfrak{p}$ and $\mathbf{r}$ are randomly generated as mentioned above.

Table 4.1 shows that the MILP formulation (4.7) that uses the vector of moments is more efficient than MILP formulation (4.21) that uses the vector of value function. However, for almost all instances the integrality gap of MILP (4.21) is lower than the one of MILP (4.7), which indicates that the linear relaxation of MILP (4.21) is in general tighter. The results also show that the valid inequalities introduced for MILP (4.7) and MILP (4.25) are efficient because adding them significantly reduces the integrality gap. In addition, Inequality 4.14 ensures that the integrality gaps of (4.7) reported in Table 4.1 are also bounds of the relative gap between v^*_{ml} and v^*_{his}.

4.5.2 Numerical experiments on instances from the literature

In this section, we evaluate the efficiency of MILP (4.7) on instances of POMDP drawn from the literature and we compare its performances with one of the state-of-the-art POMDP solver SARSOP of Kurniawati et al. [80]. In fact, SARSOP solver gives an approximate history-dependent policy for the discounted infinite horizon POMDP problem (see Remark 3). To adapt this policy for the finite horizon POMDP problem, we proceed as Dujardin et al. [40]: We compute a policy using SARSOP solver with a discount factor $\gamma = 0.999$ and we compute the expected sum of rewards over the T time steps by simulation of the history-dependent policy. We perform 10000

[1] `http://pomdp.org/examples/index.html`

(k_s, k_a)	T	$\lvert\Delta^{d}_{ml}\rvert$	Formulation	**MILP** (4.7)				**MILP** (4.21)			
				g_i (%)	g_f (%)	Opt (%)	Time (s)	g_i (%)	g_f (%)	Opt (%)	Time (s)
(3,3)	10	10^{14}	Basic	6.02	Opt	100	1.49	5.52	Opt	100	202
			Strengthened	1.70	Opt	100	0.62	0.22	Opt	100	110
	20	10^{28}	Basic	6.04	Opt	100	664	5.80	Opt	100	2621
			Strengthened	1.52	Opt	100	466	1.28	Opt	100	1735
(3,4)	10	10^{24}	Basic	9.51	0.34	87	512	8.69	3.06	43	2244
			Strengthened	3.16	0.18	87	514.4	2.16	0.79	61	1591
	20	10^{48}	Basic	9.64	1.96	43	2221	9.25	8.34	10	3431
			Strengthened	2.86	1.13	61	1731	2.38	2.03	21	3132
(3,5)	10	10^{34}	Basic	9.33	0.83	57	1591	8.65	5.49	17	2976
			Strengthened	2.35	0.38	70	1113	1.64	1.03	48	2036
	20	10^{69}	Basic	9.60	3.30	26	2702	9.23	8.62	9	3287
			Strengthened	2.14	1.14	52	1879	1.99	1.80	26	3030
(4,3)	10	10^{14}	Basic	7.39	Opt	100	26	5.63	0.39	90	520
			Strengthened	2.28	Opt	100	9.16	1.32	0.18	90	391
	20	10^{28}	Basic	6.01	1.01	60	1598	6.02	4.22	35	2407
			Strengthened	2.03	0.32	80	987	1.44	1.12	45	1995
(4,4)	10	10^{24}	Basic	12.19	0.98	65	1477	8.80	4.81	40	2178
			Strengthened	3.44	0.27	80	967	1.76	1.02	50	1888
	20	10^{48}	Basic	12.29	4.66	20	2900	9.26	8.59	20	2880
			Strengthened	3.05	1.48	30	2651	1.96	1.80	40	2270
(4,5)	10	10^{34}	Basic	11.64	1.76	35	2427	8.36	5.32	30	2636
			Strengthened	3.09	0.62	65	1345	1.74	1.32	55	1819
	20	10^{69}	Basic	12.04	5.46	5	3413	9.48	8.81	16	3031
			Strengthened	3.20	1.67	32	2490	2.15	2.08	21	3003

Table 4.1 – POMDP results using MILP (4.7) with and without (4.8), with a time limit of 3600s

simulations to compute the expectation. By definition, the objective value z_{SARSOP} obtained by using this policy is a lower bound of v^*_{his}. We run the SARSOP algorithm using the Julia library `POMDPs.jl` of Egorov et al. [43].

Instances. All the instances can be found at the link `http://pomdp.org/examples/` and further descriptions of each instance are available in the indicated literature on the same website. In particular, it contains two instances `bridge-repair` and `machine` that model maintenance problems. The first one, introduced by Ellis et al. [44], consists of the maintenance of a bridge. The modeling is almost similar to our one in Chapter 3 except that there are more available actions and they consider only one machine. Instead of just choosing whether or not to maintain the bridge, the decision maker chooses whether or not to inspect the bridge and, if so, whether or not to maintain it. The second one, introduced by Cassandra [23, Appendix H.3], consists of planning the maintenance of a machine with 4 deteriorating components. Again, the decision maker can choose to inspect before performing a maintenance of the machine. In addition, the action "maintenance" is distinguished in two different actions: repair, which consists in maintaining internal components, and replace, which consists in replacing the machine by a new one. It leads to the set of available actions $\mathcal{X}_A = \{\text{operate}, \text{inspect}, \text{repair}, \text{replace}\}$.

Metrics. We give two metrics to evaluate MILP (4.7) against the SARSOP policy. We want to compare the optimal value z^* of MILP (4.7) with the value z_{SARSOP} obtained by using the SARSOP policy. In addition, Theorem 4.4 says that z^* and z_{SARSOP} are lower bounds of v^*_{his}. We also compare these values with $z^*_{\mathrm{R^c}}$, the optimal value of the linear relaxation of MILP (4.7) with valid inequalities (4.8). By Theorem 4.4, the value of $z^*_{\mathrm{R^c}}$ is an upper bound of z^* and v^*_{his}, and consequently an upper bound of z_{SARSOP}. It leads to the relative gap $g(z) = \frac{z_{\mathrm{R^c}} - z}{z^*_{\mathrm{R^c}}}$ for any z belonging to $\{z^*, z_{\mathrm{SARSOP}}\}$.

All the results are reported in Table 4.2. The first column indicates the instance considered. The three next columns indicate respectively the cardinality of $\mathcal{X}_S$, $\mathcal{X}_O$ and $\mathcal{X}_A$ of the instance. The fourth column indicates the algorithm used. The last six columns indicate the total expected reward (Obj.) and the gap values for different finite horizon $T \in \{5, 10, 20\}$.

Numerical results. One may observe that in most cases the optimal value obtained with our MILP is close to the upper bound $z_{\mathrm{R^c}}$. Thanks to Theorem 4.4, it says that memoryless policies perform well on finite horizon for these instances. In particular, the gap is noticeably small on the instance of maintenance problem `bridge-repair`. However, as mentioned in Section 3.5, one can observe that the memoryless policies fail on instances of navigation problems [87]. We observe this phenomenon on instances of navigation problems, where the goal is to find a target in a maze, and there are a large number of states relatively to a small number of observations. It is fairly natural: using a memoryless policy in a maze is misleading because if the decision maker meets a wall, he will act as it is the first time he meets a wall, and then will always take the same actions. It seems that on these instances, the SARSOP policies work best on larger horizons, which is expected since the SARSOP policy is built for an infinite horizon problem. The results in Table 4.2 support the remark of Walraven and Spaan [158, Section 3]

	Size				Horizon					
Instances	$\lvert\mathcal{X}_S\rvert$	$\lvert\mathcal{X}_O\rvert$	$\lvert\mathcal{X}_A\rvert$	Algorithms	$T = 5$		$T = 10$		$T = 20$	
					Obj.	Gap(%)	Obj.	Gap(%)	Obj.	Gap(%)
`1d.noisy`	4	2	2	MILP	**1.56**	**18.73**	**2.97**	**19.18**	**5.82**	**18.73**
				SARSOP	0.57	70.12	0.67	81.76	0.81	88.71
`4x5x2`*	39	4	4	MILP	**0.37**	**58.13**	**0.75**	**57.45**	**1.86**	**47.58**
				SARSOP	0.08	90.87	0.08	95.28	0.08	97.50
`aircraftID`	12	5	6	MILP	**5.69**	**0.00**	**10.10**	**0.00**	**19.76**	**0.00**
				SARSOP	3.39	40.46	7.63	24.46	17.32	12.41
`aloha.10`	30	3	9	MILP	38.04	0.56	62.74	1.66	84.92	13.84
				SARSOP	**38.15**	**0.25**	**63.74**	**0.20**	**89.09**	**9.61**
`cheng.D3-1`	3	3	3	MILP	**32.29**	**1.87**	**64.38**	**1.11**	**128.55**	**0.72**
				SARSOP	32.04	2.65	64.16	1.45	128.28	0.93
`cheng.D4-1`	4	4	4	MILP	**33.83**	**5.20**	**67.37**	**4.10**	**134.45**	**3.54**
				SARSOP	32.40	9.1	65.90	6.19	133.05	4.54
`cheng.D5-1`	5	5	5	MILP	**32.89**	**3.28**	**65.64**	**2.25**	**131.12**	**1.73**
				SARSOP	32.47	4.50	65.23	2.86	130.81	1.96
`learning.c3`	24	3	12	MILP	**1.63**	**45.3**	**2.20**	**26.76**	**2.33**	**22.48**
				SARSOP	0.33	88.89	0.33	89.00	0.34	88.67
`milos-aaai97`*	20	8	6	MILP	**26.83**	**10.28**	**53.41**	**36.06**	92.09	55.06
				SARSOP	12.62	57.79	39.52	52.69	**97.73**	**52.31**
`network`*	7	2	4	MILP	20.30	2.49	95.06	22.85	203.87	36.02
				SARSOP	**20.88**	**0.00**	**95.78**	**22.26**	**245.88**	**22.98**
`bridge-repair`[a]	5	5	10	MILP	**1992.77**	**0.15**	**7801.56**	**0.44**	**27937.93**	**0.13**
				SARSOP	1514.15	24.13	6832.99	12.80	26568.42	5.03
`query.s2`	9	3	2	MILP	**21.54**	**0.95**	**46.25**	**0.10**	**96.50**	**0.11**
				SARSOP	15.77	27.50	31.68	31.56	64.91	30.66
`machine`[a]	256	16	4	MILP	**4.90**	**0.00**	**9.50**	**0.81**	**17.98**	**0.05**
				SARSOP	4.90	0.00	9.35	2.38	15.69	12.79

* Instances of navigation problem

[a] Instances of maintenance problem

Table 4.2 – Numerical results on benchmark instances. The results written in bold indicate the best value obtained for each instance.

saying that the point-based algorithms for infinite discounted POMDP, such as SARSOP, can be inefficient on finite horizon.

4.6 Bibliographical remarks

POMDP with history-dependent policies. The original POMDP problem (P_{his}) has received a lot of interest in the literature, and, the state-of-the-art algorithms solving (P_{his}) are based on two fundamental results. First, a POMDP is equivalent to a MDP in the *belief state* space [42, Theorem 4]. The belief state is the posterior probability distribution of the state given past decisions and observations and the belief state space corresponds to the unit simplex. It follows that a POMDP is MDP with a continuous state space and the Bellman's equation can be written on the belief state space (see [78, Eq. (2.15)]). The second fundamental result is that the value function of this Bellman's equation is piecewise linear and concave. It enables to derive an exact dynamic programming algorithms for POMDPs with finite [144] or infinite horizon [147]. The most important ones are the Witness algorithm [64, 86] and the Incremental Pruning algorithm [22, 168]. However, these exact algorithms become quickly intractable when the size of the spaces grows. While several heuristics use value function approximations in dynamic programming [54, 145], point-based algorithms approximate the belief state space with a finite subset and derive value iteration on its belief points [80, 120, 140, 146]. Aras et al. [6] proposed a mixed-integer programming approach giving an optimal history-dependent policy that models exactly $\mathrm{P}_{\mathrm{his}}$. Once again, solving such a program is computationally expensive even for small state spaces and small observation spaces. For more details on POMDP solutions, see for instance the surveys of Monahan [107], Ross et al. [130], Shani et al. [140] or more recently the book of Krishnamurthy [78]. In a recent work, Walraven and Spaan [158] point out the reasons why the existing state-of-the-art algorithms for solving the POMDP problem over infinite horizon with discounted rewards fail to generalize to POMDP problem over a finite horizon without discounting.

POMDP with memoryless policies. As we explained in Chapter 3, choosing a memoryless policy restricts the policy space but the resulting problem (P_{ml}) is NP-hard [87]. Littman [87] proposed a branch-and-bound heuristic that explores the policy space Δ_{ml} and Meuleau et al. [106] generalized it into an exact greedy algorithm for solving the problem with infinite horizon and discounted reward. In the reinforcement learning community, several policy iteration like algorithms have been proposed to find a stationary stochastic policy [59, 84, 143].

In the case of solving the MDP problem, the classical linear programming formulation (4.13) [38] uses the moments of the distribution as variables. This formulation is called "dual formulation" in the book of Puterman [124] because it is the dual of the well-known linear formulation of the Bellman equation for finite horizon MDP (see, e.g., [124]).

5 Integer programming for weakly coupled POMDPs

The goal of this chapter is to introduce mathematical programming formulations and algorithms to find a "good" policy for a weakly coupled POMDP ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). We recall that ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$) has been introduced in Section 3.2 and can be formulated as

$$\max_{\boldsymbol{\delta}\in\Delta_{\mathrm{ml}}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(\mathbf{S}_t,\mathbf{A}_t,\mathbf{S}_{t+1})\right] \tag{$\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$}$$

In this chapter, we denote by v_{ml}^{*} the optimal value of ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). Since a weakly coupled POMDP is a POMDP on the state space $\mathcal{X}_S$, the observation space $\mathcal{X}_O$ and the action space $\mathcal{X}_A$, we could in principle apply MILP (4.7) to solve ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). However, the number of constraints and variables grows exponentially with the number of components M, and becomes quickly intractable. Even the linear relaxation of MILP (4.7) becomes intractable. Indeed, Theorem 4.4 ensures that the linear relaxation of MILP (4.7) is equivalent to its MDP approximation. In the case of a weakly coupled POMDP, its MDP approximation is equivalent to the weakly coupled dynamic program of Adelman and Mersereau [2], which is intractable even for small values of M.

To address this issue, we introduce a new MILP formulation based on the results of Chapter 4. This formulation is an approximation of ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$) and has a tractable number of constraints and variables. The approximation is based on a relaxation of the linking constraint in the definition of the action space (3.2). To evaluate the quality of the approximation, we introduce formulations whose optimal values are lower bound and upper bound of ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$), which are also bounds of the optimal value of the integer program. Numerical experiments on medium scale instances of multi-armed bandits show that these bounds are close. In the previous chapter, MILP (4.7) gave explicitly an optimal policy. By explicitly, we mean that we had to solve only a single integer program, and its optimal solution provides the policy $\boldsymbol{\delta}$. This is no more the case here. The values taken by a solution of our integer formulation do not give policy that is a conditional probability distribution over the action space (3.2) given an observation. To address this issue, we introduce the *implicit* policies, i.e., the policies that are defined through a family of tailored optimization problems indexed by the possible history of observations and actions. At a given time step, the corresponding MILP is built, then solved and an action is retrieved from the optimal solution. These implicit policies will be used in Part III for the maintenance problem at Air France. In this chapter, we do not use a formulation with value function variables like in the previous chapter. Chapter 5 is organized as follows

- Section 5.1 introduces the MILP, which is an approximation of ($\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$).
- Section 5.2 extends the valid inequalities (4.8) of MILP (4.7) to the new MILP.
- Section 5.3 shows how the linear relaxation of our MILP is related to POMDP with history-dependent policies ($\mathrm{P}_{\mathrm{his}}$).
- Section 5.4 introduces the shared upper bound and the shared lower bound on the optimal value of the approximation and ($\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$).
- Section 5.5 shows how we build an implicit feasible policy for ($\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$) based on our integer formulation.
- Section 5.6 introduces a rolling horizon heuristic that exploits the history-dependent policy towards a practical solution.
- Section 5.7 provides several numerical experiments on Multi-armed Bandits which are partially observable.

For convenience, given a POMDP $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$ and a finite horizon T, we define respectively the feasible sets of NLP (4.1) and MILP (4.7) as $\mathcal{Q}(T, \mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p})$ and $\mathcal{Q}^{\mathrm{d}}(T, \mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p})$. We write respectively $\mathcal{Q}$ and $\mathcal{Q}^{\mathrm{d}}$ when $(T, \mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p})$ is clear from the context.

5.1 An approximate integer program

Consider a weakly coupled POMDP $\big((\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m\in[M]}, \mathbf{b}\big)$. Based on the results of Chapter 4, a naive approach is to use MILP (4.7) on the POMDP $\big(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}', \mathbf{r}'\big)$, where $\mathcal{X}_S = \mathcal{X}_S^1 \times \cdots \times \mathcal{X}_S^M$, $\mathcal{X}_O = \mathcal{X}_O^1 \times \cdots \times \mathcal{X}_O^M$, and, $\mathcal{X}_A$, $\mathfrak{p}'$, and $\mathbf{r}'$ are respectively defined by (3.2)-(3.5). As we explained in the introduction, the number of variables and the number constraints of MILP (4.7) are exponential in the number of components M, which makes its resolution or the resolution of its linear relaxation intractable. In this section, we propose an approximate integer program that breaks this curse of dimensionality.

We introduce variables $\boldsymbol{\tau}^m = \Big((\tau_s^{1,m})_s, (\tau_{sas'}^{t,m})_{s,a,s'}, (\tau_{soa}^{t,m})_{s,o,a}, (\tau_a^{t,m})_a\Big)_{t\in[T]}$, $\boldsymbol{\delta}^m = (\delta^{t,m})_{t\in[T]}$ for all m in $[M]$, and the following MILP.

$$\max_{\boldsymbol{\tau},\boldsymbol{\delta}} \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r^m(s,a,s')\tau_{sas'}^{t,m} \tag{5.1a}$$

$$\text{s.t. } \big(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m\big) \in \mathcal{Q}^{\mathrm{d}}\big(T, \mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m\big) \qquad \forall m \in [M] \tag{5.1b}$$

$$\sum_{s\in\mathcal{X}_S^m, o\in\mathcal{X}_A^m} \tau_{soa}^{t,m} = \tau_a^{t,m} \qquad \forall a \in \mathcal{X}_A^m, m\in[M], t\in[T] \tag{5.1c}$$

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a)\tau_a^{t,m} \leqslant \mathbf{b} \qquad \forall t\in[T] \tag{5.1d}$$

By Theorem 4.1, constraints (5.1b) and (5.1c) ensure that the variables $\boldsymbol{\tau}^m$ can still be interpreted as the vector of moments of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}^m}$ induced by the deterministic policy $\boldsymbol{\delta}^m$ for each component m in $[M]$. Constraint (5.1d) is a surrogate modeling of the linking constraints in the definition of $\mathcal{X}_A$ (3.2). However, given a feasible solution $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$ there is no guarantee that there exists a feasible policy $\boldsymbol{\delta}$ of ($\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$) such that the variables $(\boldsymbol{\tau}^m)_{m\in[M]}$

represent the moments of the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ on the whole system. It is the reason why we denote by $\boldsymbol{\tau}$ the approximate vector of moments instead of $\boldsymbol{\mu}$, which we used in Chapter 4.

In fact, the optimal value of MILP (5.1), denoted by z_{IP}, is neither an upper bound, nor a lower bound of v^*_{ml} in general. A feasible solution $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$ of MILP (5.1) defines a policy $\boldsymbol{\delta}^m$ on each component m in $[M]$. Unfortunately, in general, we are not able to derive from it a feasible policy of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. Indeed, for any observation $\mathbf{o}$, an element $\mathbf{a}$ satisfying $\delta^{t,m}_{a^m|o^m} = 1$ for any m in $[M]$ and t in $[T]$ does not necessary belong to $\mathcal{X}_A$. Conversely, a feasible solution $\boldsymbol{\delta}$ of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$ does not define unambiguously a deterministic policy for each component m. In Appendix A, we provide numerical examples of this phenomenon. However, z_{IP} is a good approximation for v^*_{ml} : in Section 5.4 we introduce a common lower bound and a common upper bound of z_{IP} and v^*_{ml} and numerical experiments in Section 5.7 show that these bounds turn out to be close in some practical applications.

In fact, going from MILP (4.7) to MILP (5.1) we performed three approximations:

(A) Consider "local" variables $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)$ in $\mathcal{Q}\big(T, \mathcal{X}^m_S, \mathcal{X}^m_O, \mathcal{X}^m_A, \mathfrak{p}^m\big)$ for each component m,

(B) Consider deterministic policies $\boldsymbol{\delta}^m \in \Delta^{\mathrm{d},m}$,

(C) Transform the linking constraint $\sum_{m=1}^M \mathbf{D}^m(A^m_t) \leqslant \mathbf{b}$, which is almost sure, into the constraint in expectation $\sum_{m=1}^M \mathbb{E}_{\boldsymbol{\delta}^m}\big[\mathbf{D}^m(A^m_t)\big] \leqslant \mathbf{b}$.

The size of MILP (5.1) is tractable for two reasons. First, there is a polynomial number of constraints. Indeed, the number of constraints of MILP (5.1) is $O\big(T\sum_{m=1}^M |\mathcal{X}^m_S||\mathcal{X}^m_O||\mathcal{X}^m_A|\big)$. Second, the number of binary variables in MILP (5.1) is linear in M. Indeed, the number of binary variables of MILP (5.1) is $T\sum_{m=1}^M |\mathcal{X}^m_O||\mathcal{X}^m_A|$. Furthermore, using the definition of the "local" policy set and the fact that $\mathcal{X}^m_A = \{0,1\}$ in the predictive maintenance problem with capacity constraints, a "local" policy $\boldsymbol{\delta}^m$ can be encoded using only the binary variables $\delta^{t,m}_{1|o}$ for $o \in \mathcal{X}^m_O$, $m \in [M]$ and $t \in [T]$. Hence the number of binary variables is $T\sum_{m=1}^M |\mathcal{X}^m_O|$ in that content.

5.2 Valid inequalities

Since Equalities (4.8) are valid for MILP (4.7), we can naturally derive similar equalities to tighten the linear relaxation of MILP (5.1). We introduce new variables $\tau^{t,m}_{s'a'soa}$ and the following linear inequalities.

$$\begin{aligned}
&\sum_{s'\in\mathcal{X}^m_S, a'\in\mathcal{X}^m_A} \tau^{t,m}_{s'a'soa} = \tau^{t,m}_{soa}, && \forall s \in \mathcal{X}^m_S, o \in \mathcal{X}^m_O, a \in \mathcal{X}^m_A, \\
&\sum_{a\in\mathcal{X}^m_A} \tau^{t,m}_{s'a'soa} = p^m(o|s)p^m(s|s',a')\tau^{t-1,m}_{s'a'}, && \forall s, s' \in \mathcal{X}^m_S, o \in \mathcal{X}^m_O, a' \in \mathcal{X}^m_A, \\
&\tau^{t,m}_{s'a'soa} = p^m(s|s',a',o) \sum_{\bar{s}\in\mathcal{X}^m_S} \tau^{t,m}_{s'a'\bar{s}oa}, && \forall s, s' \in \mathcal{X}^m_S, o \in \mathcal{X}^m_O, a, a' \in \mathcal{X}^m_A.
\end{aligned} \tag{5.2}$$

Proposition 5.1. *Inequalities* (5.2) *are valid for MILP* (5.1), *MILP* (5.3) *and NLP* (5.5), *and there exists solution of the linear relaxation of* (5.1) *that does not satisfy constraints* (5.2).

Proof. Proposition 4.3 ensures that equalities (5.2) are valid on each component. Hence, these inequalities are valid for MILP (5.1), MILP (5.3) and NLP (5.5). Proposition 4.3 also ensures that there are solutions of the linear relaxation of (4.7) that do not satisfy constraints (4.8) on each component. □

In practice, inequalities (5.2) help the resolution of MILP (5.1). However, since the extended formulation obtained by adding inequalities (5.2) in MILP (5.1) has a large number of variables and constraints when the number of components is large ($M \geqslant 15$), the linear relaxation takes longer to solve.

5.3 Strengths of the linear relaxation

While in Section 4.3 we showed that the linear relaxation of MILP (4.7) is equivalent to the MDP approximation, one may ask the question: How do we relate the linear relaxation of MILP (5.1) with the MDP approximation of a weakly coupled POMDP? As stated the theorem below, we are able to link the value of the linear relaxation of MILP (5.1) (with and without valid inequalities (5.2)) with the optimal value v^*_{ml} and v^*_{his}. We denote respectively by $z_{\mathrm{R^c}}$ and z_{R} the optimal values of the linear relaxations of MILP (5.1) with and without valid inequalities (5.2).

Theorem 5.2. *The linear relaxation of MILP* (5.1) *is a relaxation of the MPD approximation of the weakly coupled POMDP. Furthermore, the inequalities $v^*_{\mathrm{MDP}} \leqslant z_{\mathrm{R}}$ and $v^*_{\mathrm{his}} \leqslant z_{\mathrm{R^c}}$ hold.*

It turns out the linear relaxation of MILP (5.1) is equivalent to the fluid formulation of Bertsimas and Mišić [15], which is a relaxation of the MDP approximation of a weakly coupled POMDP.

5.4 An upper bound and a lower bound

We show that $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$ and MILP (5.1) share a lower bound z_{LB} and an upper bound z_{UB} :

$$z_{\mathrm{LB}} \leqslant v^*_{\mathrm{ml}} \leqslant z_{\mathrm{UB}}$$
$$z_{\mathrm{LB}} \leqslant z_{\mathrm{IP}} \leqslant z_{\mathrm{UB}}$$

The aim of this section is to introduce the mathematical programs that give z_{LB} and z_{UB} by playing with the approximations (A), (B) and (C). In Section 5.4.4, we propose an interpretation of the bounds obtained. The upper bound z_{UB} is difficult to compute in practice. Instead of computing z_{UB}, we propose to compute another upper bound which is based on the Lagrangian relaxation of the linking constraints (5.1d). This upper bound is much easier to compute for larger instances.

In this section, we need to compare MILP formulations that do not share the same set of variables. We therefore say that a problem P is a *relaxation* of problem P' when given a feasible solution of P' we can build a feasible solution of P with the same value.

5.4.1 The lower bound from an MILP with an exponential number of constraints

Using the same notation as in MILP (5.1), we introduce the following MILP.

$$z_{\mathrm{LB}} := \max_{\boldsymbol{\tau},\boldsymbol{\delta}} \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r_m(s,a,s')\tau_{sas'}^{t,m} \tag{5.3a}$$

$$\text{s.t. } \left(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m\right)\in\mathcal{Q}^{\mathrm{d}}\left(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\right) \qquad \forall m\in[M] \tag{5.3b}$$

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\delta_{a|o^m}^{t,m}\leqslant\mathbf{b} \qquad \forall\mathbf{o}\in\mathcal{X}_O,\ t\in[T] \tag{5.3c}$$

MILP (5.3) is obtained by using approximations (A) and (B). Note that the difference between MILP (5.1) and MILP (5.3) is that we replace the moments $\boldsymbol{\tau}^m$ in constraints (5.1d) by the policy $\boldsymbol{\delta}^m$ in constraints (5.3c). While constraints (5.1d) ensure that $\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{\delta}^m}\left[\mathbf{D}^m(A_t^m)\right]\leqslant\mathbf{b}$, constraints (5.3c) ensure that $\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{\delta}^m}\left[\mathbf{D}^m(A_t^m)|O_t^m=o^m\right]\leqslant\mathbf{b}$ for any $\mathbf{o}\in\mathcal{X}_O$. Consequently, the linking constraint of $\mathcal{X}_A$ is satisfied almost surely in MILP (5.3). We denote by z_{LB} the optimal value of MILP (5.3).

Theorem 5.3. ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$) *and MILP* (5.1) *are both relaxations of MILP* (5.3). *In particular,* $z_{\mathrm{LB}}\leqslant v_{\mathrm{ml}}^*$ *and* $z_{\mathrm{LB}}\leqslant z_{\mathrm{IP}}$.

It follows from Theorem 5.3 that MILP (5.3) gives a policy $\boldsymbol{\delta}$ that is feasible for ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$), which is interesting in itself. Theorem 5.3 tells us even more: MILP (5.3) restricts to the "decomposable" deterministic policies, i.e., policies $\boldsymbol{\delta}$ that can be written $\boldsymbol{\delta}=\prod_{m=1}^{M}\boldsymbol{\delta}^m$. However, MILP (5.3) has a exponential number of constraints, which makes it intractable for a large number of components. Indeed, there are $T\times\prod_{m=1}^{M}|\mathcal{X}_O^m|$ constraints (5.3c).

Proof of Theorem 5.3. Let $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ be a feasible solution of MILP (5.3). We prove that $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ is a feasible solution of MILP (5.1) and ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). First, we show that $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ is a feasible solution of MILP (5.1). We define the variables $\tau_a^{t,m}$ for any $a\in\mathcal{X}_A^m$, $m\in[M]$, and $t\in[T]$ such that $\sum_{s\in\mathcal{X}_S^m,o\in\mathcal{X}_O^m}\tau_{soa}^{t,m}=\tau_a^{t,m}$. In addition, we introduce the variables $\tau_o^{t,m}=\sum_{s\in\mathcal{X}_S^m,o\in\mathcal{X}_O^m}\tau_{soa}^{t,m}$ for any $o\in\mathcal{X}_O^m$, any $m\in[M]$ and $t\in[T]$. It suffices to show that inequality (5.1d) holds. We compute the left-hand side of (5.1d).

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\tau_a^{t,m}=\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m,o\in\mathcal{X}_O^m}\mathbf{D}^m(a)\delta_{a|o}^{t,m}\tau_o^{t,m}=\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a^m)\mathbb{E}_{\tau^{t,m}}[\delta_{a^m|O_t^m}^{t,m}]\leqslant\mathbf{b}$$

The first equality is a consequence of the tightness of the McCormick constraints (4.6a)-(4.6c). The second equality comes from the fact that the variables $(\tau_o^{t,m})_{o\in\mathcal{X}_O^m}$ define a probability distribution over $\mathcal{X}_O^m$. Finally, the last inequality results from

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\mathbb{E}_{\tau^{t,m}}[\delta_{a|O_t^m}^{t,m}]\leqslant\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\max_{o\in\mathcal{X}_O^m}(\delta_{a|o}^{t,m})\leqslant\mathbf{b}$$

Therefore, the inequality holds (5.1d) and MILP (5.1) is a relaxation of MILP (5.3).

Second, we show that $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ is a feasible solution of ($\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}}$). We define a policy over $\mathcal{X}_A\times\mathcal{X}_O$.

$$\delta_{\mathbf{a}|\mathbf{o}}^{t}=\prod_{m=1}^{M}\delta_{a^m|o^m}^{t,m} \tag{5.4}$$

for all $\mathbf{a} \in \mathcal{X}_A$, $\mathbf{o} \in \mathcal{X}_O$ and $t \in [T]$. It suffices to prove that $\boldsymbol{\delta}$ belongs to Δ. Let $\mathbf{o} \in \mathcal{X}_O$ and $t \in [T]$.

$$\sum_{\mathbf{a}\in\mathcal{X}_A} \delta^t_{\mathbf{a}|\mathbf{o}} = \sum_{\mathbf{a}\in\mathcal{X}_A} \prod_{m=1}^{M} \delta^{t,m}_{a^m|o^m} = \sum_{\mathbf{a}\in\mathcal{X}_A} \prod_{m=1}^{M} \delta^{t,m}_{a^m|o^m} = 1$$

The second equality comes from the fact that for any $\mathbf{a} \in \mathcal{X}_A$ such that $\sum_{m=1}^M \mathbf{D}^m(a^m) > \mathbf{b}$, $\prod_{m=1}^M \delta^{t,m}_{a^m|o^m} = 0$ because of Constraints (5.3c). Therefore, $\boldsymbol{\delta}$ is a feasible policy of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$.

Since the objective functions are the same, the inequalities $z_{\mathrm{LB}} \leqslant v^*_{\mathrm{ml}}$ and $z_{\mathrm{LB}} \leqslant z_{\mathrm{IP}}$ hold. □

5.4.2 An upper bound through a nonlinear formulation

Using the same notation as in MILP (5.1), we introduce the following NLP.

$$z^{\mathrm{UB}} := \max_{\boldsymbol{\tau},\boldsymbol{\delta}} \sum_{t=1}^{T}\sum_{m=1}^{M} \sum_{\substack{s,s'\in\mathcal{X}_S^m \\ a\in\mathcal{X}_A^m}} r_m(s,a,s')\tau^{t,m}_{sas'} \tag{5.5a}$$

$$\text{s.t. } \left(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m\right) \in \mathcal{Q}\left(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\right) \qquad \forall m \in [M] \tag{5.5b}$$

$$\sum_{s\in\mathcal{X}_S^m, o\in\mathcal{X}_A^m} \tau^{t,m}_{soa} = \tau^{t,m}_{a} \qquad \forall a \in \mathcal{X}_A^m, m\in[M], t\in[T] \tag{5.5c}$$

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a)\tau^{t,m}_a \leqslant \mathbf{b} \qquad \forall t\in[T] \tag{5.5d}$$

NLP (5.5) is obtained by using approximations (A) and (C). Once again, variables $\boldsymbol{\tau}^m$ can be interpreted as the vector of moments of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}^m}$ on component m but there is no guarantee that it defines a joint probability distribution over the whole system. However, Theorem 5.4 ensures that it gives a relaxation of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. We denote by z_{UB} the optimal value of NLP (5.5).

Theorem 5.4. *NLP* (5.5) *is a relaxation of* $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$ *and MILP* (5.1). *In particular,* $v^*_{\mathrm{ml}} \leqslant z_{\mathrm{UB}}$ *and* $z_{\mathrm{IP}} \leqslant z_{\mathrm{UB}}$.

NLP (5.5) is a Quadratically Constrained Quadratic Program (QCQP) due to constraints (5.5b) and is in general non-convex. Hence, it is hard to solve NLP (5.5) in practice because it requires to use a Spatial Branch-and-Bound However, the number of variables and the number of constraints are polynomial. Thanks to the recent advances of QCQP solvers such as `Gurobi 9.0` [52], we are able to solve NLP (5.5) to optimality in a reasonable computation time for small instances. For larger instances, the solver is no longer efficient because it reaches the limits of a Spatial Branch-and-Bound [85].

Proof of Theorem 5.4. First, we prove that NLP (5.5) is a relaxation of MILP (5.1). Let $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ be a feasible solution of MILP (5.1). We prove that $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$ is a feasible solution of NLP (5.5). It suffices to prove that for all $m \in [M]$, $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)$ satisfies constraints (4.1g). It comes from the tightness of the McCormick inequalities (4.6a)-(4.6c) when the policy is deterministic. Hence, it is a relaxation with the same objective function. Therefore, the inequality $z_{\mathrm{IP}} \leqslant z_{\mathrm{UB}}$ holds.

Second, we prove that NLP (5.5) is a relaxation of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. Let $\boldsymbol{\delta}$ be a feasible policy of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. We want to define a solution of the non-linear program (5.5). We extend the domain of $\boldsymbol{\delta}$ to $\mathcal{X}_A$ by setting $\delta^t_{\mathbf{a}|\mathbf{o}} = 0$ when $\sum_{m\in[M]} \mathbf{D}^m(a^m) > \mathbf{b}$, for all $\mathbf{o} \in \mathcal{X}_O$. It is easy to see that $\boldsymbol{\delta}$ is still a policy in $\mathcal{X}_A$. Theorem 4.1 ensures that there exists $\boldsymbol{\mu}$ such that $(\boldsymbol{\mu}, \boldsymbol{\delta})$ is a feasible solution of MILP (4.7). We define the variables $\boldsymbol{\tau}^m$ on component $m \in [M]$ by induction

$$\tau^{1,m}_s = \sum_{s^{-m}\in\mathcal{X}_S^{-m}} \mu^1_{\mathbf{s}}, \qquad \tau^{1,m}_{soa} = \Bigg(\sum_{\substack{s^{-m}\in\mathcal{X}_S^{-m}\\ o^{-m}\in\mathcal{X}_O^{-m}\\ a^{-m}\in\mathcal{X}_A^{-m}}} \delta^t_{\mathbf{a}|\mathbf{o}} \prod_{m'\neq m} p^{m'}(o^{m'}|s^{m'})\tau^{1,m'}_{s^{m'}} \Bigg) p^m(o|s)\tau^{1,m}_s,$$

$$\tau^{t,m}_{sas'} = \sum_{o'\in\mathcal{X}_O^m, a'\in\mathcal{X}_A^m} \tau^{t,m}_{s'o'a'}, \qquad \tau^{t+1,m}_{soa} = \Bigg(\sum_{\substack{s^{-m}\in\mathcal{X}_S^{-m}\\ o^{-m}\in\mathcal{X}_O^{-m}\\ a^{-m}\in\mathcal{X}_A^{-m}}} \delta^t_{\mathbf{a}|\mathbf{o}} \prod_{m'\neq m} p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}} \Bigg) p^m(o|s) \sum_{s'\in\mathcal{X}_S^m, a'\in\mathcal{X}_A^m} \tau^{t,m}_{s'a's},$$

and the policy $\boldsymbol{\delta}^m$

$$\delta^{t,m}_{a|o} = \sum_{\substack{s^{-m}\in\mathcal{X}_S^{-m}\\ o^{-m}\in\mathcal{X}_O^{-m}\\ a^{-m}\in\mathcal{X}_A^{-m}}} \delta^t_{\mathbf{a}|\mathbf{o}} \prod_{m'\neq m} p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}},$$

for all $a \in \mathcal{X}_A^m$, $o \in \mathcal{X}_O^m$ and $t \in [T]$. By definition of $\boldsymbol{\tau}^m$, if $\boldsymbol{\delta}^m$ is in Δ^m, then the constraints (5.5b) are satisfied by $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$. We prove that $\boldsymbol{\delta}^m$ is in Δ^m_{ml}.

$$\sum_{a\in\mathcal{X}_A^m} \delta^{t,m}_{a|o} = \sum_{a\in\mathcal{X}_A^m} \sum_{\substack{s^{-m}\in\mathcal{X}_S^{-m}\\ o^{-m}\in\mathcal{X}_O^{-m}\\ a^{-m}\in\mathcal{X}_A^{-m}}} \delta^t_{\mathbf{a}|\mathbf{o}} \prod_{m'\neq m} p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}} = \sum_{\substack{s^{-m}\in\mathcal{X}_S^{-m}\\ o^{-m}\in\mathcal{X}_O^{-m}}} \prod_{m'\neq m} p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}} = 1$$

for all $o \in \mathcal{X}_O^m$, $m \in [M]$ and $t \in [T]$. The last equality comes from the fact that by induction we have that $\sum_{s\in\mathcal{X}_S^m} \tau^{t,m}_s = 1$. Therefore, $\boldsymbol{\delta}^m \in \Delta^m_{\mathrm{ml}}$.

It remains to prove that constraints (5.5d) are satisfied by $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$. We compute the left-hand side of constraint (5.5d).

$$\begin{aligned}
\sum_{m=1}^M \sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a)\tau^{t,m}_a &= \sum_{m=1}^M \sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a) \Bigg(\sum_{\substack{\mathbf{s}\in\mathcal{X}_S,\mathbf{o}\in\mathcal{X}_O\\ \mathbf{a}'\in\mathcal{X}_A: a'^m = a^m}} \delta^t_{\mathbf{a}'|\mathbf{o}} \prod_{m'=1}^M p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}} \Bigg) \\
&= \sum_{\substack{\mathbf{s}\in\mathcal{X}_S,\mathbf{o}\in\mathcal{X}_O\\ \mathbf{a}'\in\mathcal{X}_A}} \delta^t_{\mathbf{a}'|\mathbf{o}} \prod_{m'=1}^M p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}} \sum_{m=1}^M \sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a^m) \\
&= \overbrace{\sum_{\substack{\mathbf{s}\in\mathcal{X}_S,\mathbf{o}\in\mathcal{X}_O\\ \mathbf{a}\in\mathcal{X}_A}} \delta^t_{\mathbf{a}|\mathbf{o}} \prod_{m'=1}^M p^{m'}(o^{m'}|s^{m'})\tau^{t,m'}_{s^{m'}}}^{=1} \underbrace{\sum_{m=1}^M \mathbf{D}^m(a^m)}_{\leqslant \mathbf{b}} \\
&\leqslant \mathbf{b}
\end{aligned}$$

Therefore, constraints (5.5d) are satisfied. Consequently, NLP (5.5) is a relaxation of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. In addition, the objective functions are equal. We deduce that $v^*_{\mathrm{ml}} \leqslant z_{\mathrm{UB}}$. □

5.4.3 A tractable upper bound through Lagrangian relaxation

When the number of components increases, computing the upper bound z_{UB} becomes quickly intractable. Fortunately, we can obtain a weaker upper bound of v^*_{ml} and z_{IP} that is more tractable to compute. To do so, we use the Lagrangian relaxation technique on constraints (5.5d) of NLP (5.5). This technique has already been used in the literature on weakly coupled dynamic programs to compute upper bounds [2, 55, 165].

We denote by $\boldsymbol{\beta} = (\beta^t)_{t\in[T]}$ the dual variables associated with constraints (5.5d). If we relax constraints (5.5d), then we obtain the Lagrangian function We introduce a collection of non-negative variables $\boldsymbol{\beta} = (\beta^t)_{t\in[T]}$ with $\beta^t \in \mathbb{R}^q_+$ for any $t \in [T]$ and the following Lagrangian function

$$\mathcal{L}(\boldsymbol{\tau},\boldsymbol{\delta},\boldsymbol{\beta}) = \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r^m(s,a,s')\tau^{t,m}_{sas'} + \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\left(\mathbf{b} - \sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\tau^{t,m}_a\right),$$

for any $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$. Then, we introduce the dual function $\mathcal{G}:\mathbb{R}_+^{T\times q}\to\mathbb{R}$, with values

$$\begin{aligned}\mathcal{G}(\boldsymbol{\beta}) := \max_{\boldsymbol{\tau},\boldsymbol{\delta}}\ & \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r^m(s,a,s')\tau^{t,m}_{sas'} + \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\left(\mathbf{b} - \sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\tau^{t,m}_a\right) \\ \text{s.t. }\ & (\boldsymbol{\tau}^m,\boldsymbol{\delta}^m) \in \mathcal{Q}(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m) \quad \forall m\in[M]\end{aligned} \tag{5.6}$$

By weak duality, for any $\boldsymbol{\beta}$, the dual function (5.6) provides an upper bound obtained by using Approximation (A).

We now explain how to compute the dual function. As it is usually the case for Lagrangian relaxation, for every $\boldsymbol{\beta}\in\mathbb{R}_+^{T\times q}$, the maximum in the computation of $\mathcal{G}(\boldsymbol{\beta})$ decomposes over the sum of the maximum over each component. However, the formulations obtained for each component are still nonlinear. Fortunately, the following proposition ensures that we can linearize the formulation without changing the value of the dual function.

Proposition 5.5. *For all $\boldsymbol{\beta}\in\mathbb{R}_+^{T\times q}$, the dual function can be written as*

$$\mathcal{G}(\boldsymbol{\beta}) = \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\mathbf{b} + \sum_{m=1}^{M}\mathcal{G}^m(\boldsymbol{\beta}) \tag{5.7}$$

where $\mathcal{G}^m(\boldsymbol{\beta})$ is the quantity

$$\begin{aligned}\mathcal{G}^m(\boldsymbol{\beta}) := \max_{\boldsymbol{\tau}^m,\boldsymbol{\delta}^m}\ & \sum_{t=1}^{T}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} \left(r^m(s,a,s') - (\boldsymbol{\beta}^t)^{\mathbf{T}}\mathbf{D}^m(a)\right)\tau^{t,m}_{sas'} \\ \text{s.t. }\ & (\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)\in\mathcal{Q}^{\mathrm{d}}(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m)\end{aligned}$$

Proof. Let $\boldsymbol{\beta} \in \mathbb{R}_+^{Tq}$. Then, the value function $\mathcal{G}$ in $\boldsymbol{\beta}$ can be written:

$$\begin{aligned} \mathcal{G}(\boldsymbol{\beta}) = \max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}} \quad & \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{\substack{s,s'\in\mathcal{X}_S^m \\ a\in\mathcal{X}_A^m}} \Big(r^m(s,a,s') - (\boldsymbol{\beta}^t)^{\mathbf{T}}\mathbf{D}^m(a)\Big)\tau_{sas'}^{t,m} + \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\mathbf{b} \\ \text{s.t.} \quad & \big(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m\big) \in \mathcal{Q}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big) \qquad \forall m\in[M] \end{aligned}$$

Since the second term does not depend on $(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}$, we only consider the maximization on the first term. In such a problem, there are no linking constraints between the components, which enables to decompose the maximization operator along the components as follows.

$$\begin{aligned} \mathcal{G}(\boldsymbol{\beta}) = \sum_{m=1}^{M}\max_{\boldsymbol{\tau}^m,\boldsymbol{\delta}^m} \quad & \sum_{t=1}^{T}\sum_{\substack{s,s'\in\mathcal{X}_S^m \\ a\in\mathcal{X}_A^m}} \Big(r^m(s,a,s') - (\boldsymbol{\beta}^t)^{\mathbf{T}}\mathbf{D}^m(a)\Big)\tau_{sas'}^{t,m} + \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\mathbf{b} \\ \text{s.t.} \quad & \big(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m\big) \in \mathcal{Q}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big) \end{aligned}$$

Theorem 4.1 ensures that the optimization subproblem above on component m corresponds to a POMDP problem with memoryless policies of POMDP $\big(\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m,\tilde{\mathbf{r}}\big)$ where $\tilde{r}^m(s,a,s') = r^m(s,a,s') - (\boldsymbol{\beta}^t)^{\mathbf{T}}\mathbf{D}^m(a)$ for any $s,s'\in\mathcal{X}_S^m$ and $a\in\mathcal{X}_A^m$. Thanks to Proposition 4.2, the subproblem on component m can be solved using deterministic policies. Therefore, we can replace $\mathcal{Q}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big)$ by $\mathcal{Q}^{\mathrm{d}}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big)$ for any component m and we obtain

$$\mathcal{G}(\boldsymbol{\beta}) = \sum_{t=1}^{T}(\beta^t)^{\mathbf{T}}\mathbf{b} + \sum_{m=1}^{M}\mathcal{G}^m(\boldsymbol{\beta}),$$

which achieves the proof. □

It follows from Proposition 5.5 that the dual function can be computed by solving MILP (4.7) on each component of the system, which is in general easier than solving NLP (5.5). As stated in the following proposition, it gives us an upper bound that is not worse than the optimal value of the linear relaxation of MILP (5.1). We denote respectively by z_{R} and $z_{\mathrm{R^c}}$ the optimal values of the linear relaxation of MILP (5.1) and the linear relaxation of MILP (5.1) with valid inequalities (5.2), and we define the Lagrangian relaxation $z_{\mathrm{LR}} := \min_{\boldsymbol{\beta}\in\mathbb{R}_+^{Tq}}\mathcal{G}(\boldsymbol{\beta})$.

Proposition 5.6. *The value of the Lagrangian relaxations of MILP* (5.1) *and NLP* (5.5) *are equal and the following inequalities hold:*

$$z_{\mathrm{UB}} \leqslant z_{\mathrm{LR}} \leqslant z_{\mathrm{R^c}} \leqslant z_{\mathrm{R}}$$

Proof. Thanks to Proposition 5.5, the dual functions of MILP (5.1) and NLP (5.5) are equal. It follows that the value of the Lagrangian relaxations are equal.

First, the inequality $z_{\mathrm{UB}} \leqslant z_{\mathrm{LR}}$ comes from weak duality (see e.g. Bertsekas [13, Proposition 5.1.3]). Second, to show the second inequality $z_{\mathrm{LR}} \leqslant z_{\mathrm{R}}$, it suffices to observe that the dual function $\mathcal{G}(\boldsymbol{\beta})$ of NLP (5.5) is also the dual function of MILP (5.1). Indeed, in the expression of $\mathcal{G}^m(\boldsymbol{\beta})$ we can replace $\mathcal{Q}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big)$ by $\mathcal{Q}^{\mathrm{d}}\big(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m\big)$ because the re always exists an optimal policy that is deterministic on the POMDP $\big(\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m,\mathbf{r}^m - \boldsymbol{\beta}^T\mathbf{D}^m\big)$. A classical result in operations research (see e.g. Geoffrion [48, Theorem 1]) states that the bound of the Lagrangian relaxation of an integer program is not worse than the bound of the linear relaxation. It shows that $z_{\mathrm{LR}} \leqslant z_{\mathrm{R}}$.

It remains to prove that $z_{\mathrm{LR}} \leqslant z_{\mathrm{R^c}}$ and $z_{\mathrm{R^c}} \leqslant z_{\mathrm{R}}$. The second one comes from the fact that we have a smaller feasible set in the linear relaxation by adding the valid inequalities. The first one comes by adding valid inequalities (5.2) in the expression of $\mathcal{G}^m(\boldsymbol{\beta})$, which is possible since the inequalities are valid, and by using the same arguments (weak duality and Geoffrion's Theorem) we conclude that $z_{\mathrm{LR}} \leqslant z_{\mathrm{R^c}}$. □

Now we propose two methods to compute the value of the Lagrangian relaxation z_{LR}: the subgradient algorithm and a column generation algorithm. Depending on the user preferences, both approaches have their advantages and drawbacks. In this thesis, we choose to use the column generation algorithm, which turns to be efficient on numerical experiments (see Section 5.7).

Subgradient algorithm. Since the dual function $\mathcal{G}(\boldsymbol{\beta})$ is convex in $\boldsymbol{\beta}$, we can apply the classical subgradient methods (see e.g. Bertsekas [13]), which is known to converge to $\min_{\boldsymbol{\beta}\in\mathbb{R}_+^{Tq}} \mathcal{G}(\boldsymbol{\beta})$ for a step length carefully chosen. Note that each step requires to solve M times MILP (5.1) to evaluate the dual function. Even if the convergence to the optimum of such an algorithm can be slow, we obtain quickly an upper bound that is tighter than z_{R} and $z_{\mathrm{R^c}}$.

Column generation algorithm. Thanks to Geoffrion's Theorem [29, Theorem 8.2], the value of the Lagrangian relaxation MILP (5.1) satisfies the following Dantzig-Wolfe reformulation:

$$\begin{aligned} z_{\mathrm{LR}} = \max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}} \quad & \sum_{m=1}^{M}\sum_{t=1}^{T}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r^m(s,a,s')\tau_{sas'}^{t,m} \\ \text{s.t.} \quad & (\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)\in\mathrm{Conv}\Big(\mathcal{Q}^{\mathrm{d}}(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m)\Big), \quad && \forall m\in[M] \\ & \sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m}\mathbf{D}^m(a)\tau_a^{t,m}\leqslant \mathbf{b}, && \forall t\in[T], \end{aligned} \tag{5.11}$$

where we included the constraints $\sum_{s\in\mathcal{X}_S^m, o\in\mathcal{X}_A^m}\tau_{soa}^{t,m} = \tau_a^{t,m}$ in the set $\mathcal{Q}^{\mathrm{d}}(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m)$ and $\mathrm{Conv}(X)$ denotes the convex hull of a set X. We can compute this formulation using an exact column generation algorithm.

Using the definition of the convex hull, we can reformulate MILP (5.11) as the following master problem:

$$z_{\mathrm{LR}} = \max_{(\boldsymbol{\omega}^m,\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}} \sum_{m=1}^{M}\sum_{t=1}^{T}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} r^m(s,a,s')\tau_{sas'}^{t,m} \tag{5.12a}$$

$$\text{s.t. } (\boldsymbol{\tau}^m,\boldsymbol{\delta}^m) = \sum_{(\boldsymbol{\tau},\boldsymbol{\delta})\in\mathcal{Q}^{\mathrm{d},m}} \omega^m_{\boldsymbol{\tau},\boldsymbol{\delta}}(\boldsymbol{\tau},\boldsymbol{\delta}) \qquad \forall m\in[M] \tag{5.12b}$$

$$\sum_{(\boldsymbol{\tau},\boldsymbol{\delta})\in\mathcal{Q}^{\mathrm{d},m}} \omega^m_{\boldsymbol{\tau},\boldsymbol{\delta}} = 1 \qquad \forall m\in[M] \tag{5.12c}$$

$$\sum_{m=1}^{M}\sum_{a\in\mathcal{X}_A^m} \mathbf{D}^m(a)\tau_a^{t,m} \leqslant \mathbf{b} \qquad \forall t\in[T] \tag{5.12d}$$

$$\omega^m_{\boldsymbol{\tau},\boldsymbol{\delta}} \geqslant 0 \qquad \forall(\boldsymbol{\tau},\boldsymbol{\delta})\in\mathcal{Q}^{\mathrm{d},m}, \forall m\in[M] \tag{5.12e}$$

where $\mathcal{Q}^{\mathrm{d},m} = \mathcal{Q}^{\mathrm{d}}(T,\mathcal{X}_S^m,\mathcal{X}_O^m,\mathcal{X}_A^m,\mathfrak{p}^m)$ for every m in $[M]$. It follows that the pricing subproblem on component m writes down:

$$\begin{aligned} z^m := \max_{(\boldsymbol{\tau},\boldsymbol{\delta})} &\sum_{t=1}^{T}\sum_{\substack{s,s'\in\mathcal{X}_S^m\\ a\in\mathcal{X}_A^m}} \big(r^m(s,a,s') - \beta_t^{\mathrm{T}}\mathbf{D}^m(a)\big)\tau_{sas'}^t \\ \text{s.t. } &(\boldsymbol{\tau},\boldsymbol{\delta})\in\mathcal{Q}^{\mathrm{d},m}, \end{aligned} \tag{5.13}$$

where $\boldsymbol{\beta} = (\beta^t)_{t\in[T]} \in \mathbb{R}_+^{Tq}$ is the vector dual variables of the linking constraint (5.12d). We denote by $\boldsymbol{\pi} = (\pi^m)_{m\in[M]}$ the vector of dual variables of Constraints (5.12c). It follows that the reduced cost can be written $c = \sum_{m=1}^{M} z^m + \pi^m$.

Now we are able to derive the column generation algorithm. We assume that $\mathcal{X}_A \neq \emptyset$ (otherwise the decision maker cannot choose any action). Hence, there exists at least one element $\mathbf{a} \in \mathcal{X}_A^1 \times \cdots \times \mathcal{X}_A^M$ such that $\sum_{m=1}^{M}\mathbf{D}^m(a^m) \leqslant \mathbf{b}$. Let $\mathbf{a}_{\mathrm{e}}$ be such an element in $\mathcal{X}_A$.

Algorithm 1 returns an optimal solution of the master problem (5.12) and the value of the Lagrangian relaxation z_{LR}. We omit the proof in this thesis.

5.4.4 Interpretation of the bounds

In this section, we summarize the links between the feasible sets of MILP (5.3), MILP (5.1), NLP (5.5) and $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. It enables to give a probabilistic interpretation of the bounds z_{LB} and z_{UB}. Figure 5.1 illustrates Theorems 5.3, 5.4 and the Lagrangian relaxation property.

First, MILP (5.3) corresponds to a restriction of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$ because we choose policies that are deterministic on each component and all the actions induced by the feasible policy satisfies the linking constraints (3.2) almost surely.

Second, NLP (5.5) allows stochastic policies, and constraint (5.5b) ensures that the linking constraints in (3.2) are satisfied in expectation. Therefore, z_{LB} and z_{UB} are respectively a "deterministic" approximation and a "in expectation" approximation of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$.

Algorithm 1 Column generation to compute z_{LR}.

```
1: Input: $T$, weakly coupled POMDP $\big((\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m\in[M]}, \mathbf{b}\big)$
2: Output: An optimal solution of the master problem (5.12) and the value $z_{\mathrm{LR}}$
3: Initialize $z \leftarrow -\infty$, $\mathcal{Q}'^m \leftarrow \emptyset$
4: for $m = 1, \dots, M$ do
5:     Define $\boldsymbol{\delta}^m$ such that $\boldsymbol{\delta}_{a|o}^{t,m} = \mathbb{1}_{a_e^m}(a)$ for every $a \in \mathcal{X}_A^m$, $o \in \mathcal{X}_O^m$ and $t \in [T]$
6:     Compute $\boldsymbol{\tau}^m$ such that $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m) \in \mathcal{Q}^{\mathrm{d},m}$
7:     $z^m \leftarrow \infty$ and $\pi^m \leftarrow \infty$
8: end for
9: while $\sum_{m=1}^M z^m + \pi^m > 0$ do
10:    Add column: $\mathcal{Q}'^m \leftarrow \mathcal{Q}'^m \cup \{(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)\}$
11:    Solve master problem (5.12) restricted to $(\mathcal{Q}'^m)_{m\in[M]}$ to obtain dual variables $(\boldsymbol{\beta}, \boldsymbol{\pi})$
12:    $z \leftarrow$ Optimal value of the restricted master problem
13:    for $m = 1, \dots, M$ do
14:        Set reward $\tilde{r}_t^m(s,a,s') := r^m(s,a,s') - \beta_t^{\mathbf{T}} \mathbf{D}^m(a)$ for every $s, s' \in \mathcal{X}_S^m$ and $a \in \mathcal{X}_A^m$
15:        Solve MILP (4.7) with valid inequalities (4.8) for POMDP $(\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}, \tilde{\mathbf{r}})$ to obtain
           $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)$ and $z^m$
16:    end for
17: end while
18: Set $\boldsymbol{\tau}^m := \sum_{(\boldsymbol{\tau}', \boldsymbol{\delta}') \in \mathcal{Q}'^m} \omega_{\boldsymbol{\tau}', \boldsymbol{\delta}'}^m \boldsymbol{\tau}'$ for every $m \in [M]$
19: return $(\boldsymbol{\tau}^m)_{m\in[M]}$ and $z$
```

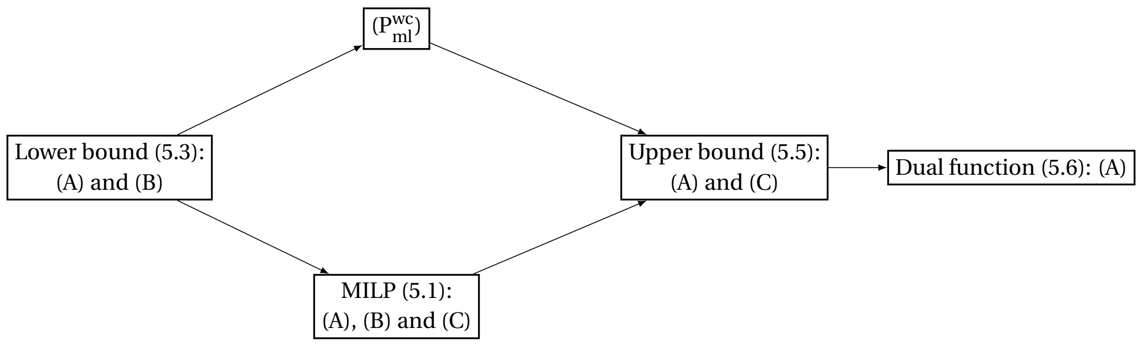

(P$^{\mathrm{wc}}_{\mathrm{ml}}$)
Lower bound (5.3): (A) and (B)
Upper bound (5.5): (A) and (C)
Dual function (5.6): (A)
MILP (5.1): (A), (B) and (C)

Figure 5.1 – The relaxations of Section 5.4. An arrow from Problem X to a Problem Y indicates that Y is a relaxation of X in the sense we defined at the beginning of this section. In each block, we indicate which assumptions (see p. 61) we use to obtain the formulations.

5.4.5 Benefits and drawbacks of the formulations

Figure 5.1 summarizes the links between the feasible sets of the different formulations they have been established in the theorems. Table 5.1 highlights the benefits and the drawbacks of the different formulations. It reports the behavior of the formulations regarding several criteria formulated as questions: are the numbers of variables (Pol. variables) and constraints (Pol. constraints) polynomial? Does the formulation have linking constraints between the components (Link. constraints)? Is the formulation linear (Linearity)? Are there integer variables in the formulation (Int. variables)? Does the formulation provide a feasible policy (Feas. pol.)? Is the optimal value an upper bound or a lower bound regarding v^*_{ml}? Is the formulation tractable regarding the size of the instances (small with $|\mathcal{X}_S| \leqslant 10^2$, medium with $10^2 \leqslant |\mathcal{X}_S| \leqslant 10^4$ and large with $|\mathcal{X}_S| \geqslant 10^4$)? The tractability criteria should be understood as an advice on the formulation to choose and the scale order is only one indicator among others.

									Tractability		
Formulations	Pol. variables	Pol. constraints	Link. constraints	Linearity	Int. variables	Feas. policy	Lower bound	Upper bound	Small	Medium	Large
MILP (5.1)	✓	✓	✓	✓	✓				✓	✓	
Lower bound (5.3)	✓		✓	✓	✓	✓	✓		✓		
Upper bound (5.5)	✓	✓	✓					✓	✓		
Lagrangian Relaxation	✓	✓		✓	✓			✓	✓	✓	✓
Linear Relaxation of (5.1)	✓	✓	✓	✓				✓	✓	✓	✓

Table 5.1 – Comparison of the properties of the formulations.

All the formulations we propose have a polynomial number of variables. Each formulation has benefits and drawbacks. While MILP (5.3) is an MILP with an exponential number of constraints, NLP (5.5) is a NLP with a polynomial number of constraints. The presence of linking constraints between the components in the MILP (5.3), MILP (5.1), and the NLP (5.5), increases the difficulty to solve a linear or nonlinear formulation. Indeed, these constraints are also called "complicating constraints" [28] because it prevents of solving a single POMDP problem on each component individually. Since there are no linking constraints in the integer program (5.7) that computes the dual function, it is the easiest to solve.

5.5 Deducing an history-dependent policy from MILP (5.1)

In MILP (5.1), we consider "local" policies $\boldsymbol{\delta}^m$ on each component m in $[M]$. However, in general, given a vector of "local" policies $(\boldsymbol{\delta}^m)_{m\in[M]}$, there is no guarantee that there exists a policy $\boldsymbol{\delta}$ that coincides with $\boldsymbol{\delta}^m$ for every components m in $[M]$. In this section we describe how we build an implicit policy that is feasible for $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{his}})$. We recall that an implicit policy of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{his}})$ is a policy $\boldsymbol{\delta}$ such that each value $\delta^t_{\mathbf{a}|\mathbf{h}}$ is computed using an tailored optimization problem. Consider a weakly coupled POMDP $(\mathcal{X}^m_S, \mathcal{X}^m_O, \mathcal{X}^m_A, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m\in[M]}, \mathbf{b})$ and a history $\mathbf{h} = (\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)$ available at time t. Conditionally to $\mathbf{h}$, the vectors of state component $\big(S^m_{t'}\big)_{1\leqslant t'\leqslant t}$ for all m in $[M]$ become independent, i.e.,

$$\mathbb{P}_{\boldsymbol{\delta}}(\mathbf{S}_t = \mathbf{s}|\mathbf{H}_t = \mathbf{h}) = \prod_{m=1}^{M} \overbrace{\mathbb{P}_{\boldsymbol{\delta}}\big(S^m_t = s^m | H^m_t = h^m\big)}^{p^m(s^m|h^m)}.$$

In the POMDP literature, the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}\big(S_t^m | H_t^m\big)$ is the belief state of component m. We can use the *belief state update* (see Remark 7) on each of the components to compute the belief state $p^m(s^m|h^m)$. We introduce the following algorithm:

Algorithm 2 History-dependent policy $\mathrm{Act}_{\mathrm{T}}^{\mathrm{t}}(\mathbf{h})$

```
1: Input An history of observations and actions $\mathbf{h} \in (\mathcal{X}_O \times \mathcal{X}_A)^{t-1} \times \mathcal{X}_O$
2: Output An action $\mathbf{a} \in \mathcal{X}_A$
3: Compute the belief state $p^m(s|h^m)$ according to the belief state update (see remark below)
   for every state $s$ in $\mathcal{X}_S^m$ and every component $m$ in $[M]$
4: Remove constraints and variables indexed by $t' < t$ in MILP (5.1) and solve the resulting
   problem with horizon $T-t$, initial probability distributions $\big(p^m(s|h^m)\big)_{s\in\mathcal{X}_S^m}$ for every com-
   ponent $m$ in $[M]$ and initial observation $\mathbf{o}$ (see Remark 4) to obtain an optimal solution
   $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$
5: Return $\mathbf{a} = (a^1, \dots, a^M)$ for which $\delta_{a^m|o^m}^{t,m} = 1$ for every $m$ in $[M]$
```

Then we define the implicit history-dependent policy $\boldsymbol{\delta}^{\mathrm{IP}}$ as follows:

$$\delta_{\mathbf{a}|\mathbf{h}}^{t,\mathrm{IP}} = \begin{cases} 1, & \text{if } \mathbf{a} = \mathrm{Act}_T^t(\mathbf{h}) \\ 0, & \text{otherwise} \end{cases}, \quad \forall \mathbf{h} \in \mathcal{X}_H^t, \mathbf{a} \in \mathcal{X}_A, t \in [T] \tag{5.14}$$

It is not clear that the policy (5.14) is a feasible policy of $(\mathrm{P}_{\mathrm{his}}^{\mathrm{wc}})$ because it is not immediate to see that the action returned by Algorithm 2 belongs to $\mathcal{X}_A$. The theorem below ensures that the implicit policy (5.14) is a feasible policy of $(\mathrm{P}_{\mathrm{his}}^{\mathrm{wc}})$ giving a higher expected reward than z_{IP}. We denote by v_{IP} the total expected reward induced by policy $\boldsymbol{\delta}^{\mathrm{IP}}$.

Theorem 5.7. *The implicit policy* $\boldsymbol{\delta}^{\mathrm{IP}}$ *defined in* (5.14) *is a feasible policy of* $(\mathrm{P}_{\mathrm{his}}^{\mathrm{wc}})$ *and the inequality* $z_{\mathrm{IP}} \leqslant v_{\mathrm{IP}} \leqslant v_{\mathrm{his}}^*$ *holds.*

Theorem 5.7 has a strong interest in practice because it enables to exploit the implicit policy (5.14) in an rolling horizon heuristic, which we describe in Section 5.6. Theorem 5.7 enables to deduce a feasible policy of $(\mathrm{P}_{\mathrm{his}}^{\mathrm{wc}})$ from MILP (5.1).

Remark 7. Given a POMDP $(\mathcal{X}_S, \mathcal{X}_O, \mathcal{X}_A, \mathfrak{p}, \mathbf{r})$, at each time t, the belief state $(p(s|H_t))_{s\in\mathcal{X}_S}$ is a sufficient statistic of the history of actions and observations H_t [42, Theorem 4]. Given the action a_t taken at time t an the observation o_{t+1} received at time $t+1$, the belief state can be easily updated over time according to the belief state update [86, Eq. (1)]:

$$p(s_{t+1}|H_{t+1}) = p(s_{t+1}|H_t, a_t, o_{t+1}) = \sum_{s\in\mathcal{X}_S} \frac{p(o_{t+1}|s)p(s|s_t, a_t)}{\sum_{s'\in\mathcal{X}_S} p(o_{t+1}|s')p(s'|s_t, a_t)} p(s|H_t)$$

△

Proof of Theorem 5.7. It suffices to prove that at each time $t \in [T]$, for every observation $\mathbf{h} \in \mathcal{X}_H^t$ the element $\mathrm{Act}_{\mathrm{T}}^{\mathrm{t}}(\mathbf{h})$ belongs to $\mathcal{X}_A$, i.e., $\sum_{m=1}^M \mathbf{D}^m\big(\mathrm{Act}_{\mathrm{T}}^{\mathrm{t,m}}(\mathbf{h})\big) \leqslant \mathbf{b}$. Let $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$ be a feasible solution of MILP (5.1) at step 4. Since $O_t^m = o_t^m$ almost surely, $\tau_{soa}^{t,m}$ is equal to 0 when $o \neq o_t^m$.

Hence, we obtain
$$\tau_a^{t,m} = \sum_{s \in \mathcal{X}_S^m, o \in \mathcal{X}_O^m} \tau_{soa}^{t,m} = \sum_{s \in \mathcal{X}_S^m} \tau_{s o_t^m a}^{t,m} = \delta_{a|o_t^m}^{t,m}$$

It ensures that $\tau_a^{t,m} \in \{0,1\}$ for any $a \in \mathcal{X}_A^m$ and $m \in [M]$. Let $\mathbf{a}^*$ be the action taken at step 5. Therefore, $\tau_a^{t,m} = 1$ when $a = a^{m*}$ and 0 otherwise. Now we compute the linking constraint of (3.2).
$$\sum_{m=1}^{M} \mathbf{D}^m(a^{m*}) = \sum_{m=1}^{M} \mathbf{D}^m(a^{m*})\tau_{a^{m*}}^{t,m} = \sum_{m=1}^{M} \sum_{a \in \mathcal{X}_A^m} \mathbf{D}^m(a)\tau_a^{t,m} \leqslant \mathbf{b}$$

The last inequality comes from the fact that $(\boldsymbol{\tau}^m, \boldsymbol{\delta}^m)_{m\in[M]}$ satisfies constraint (5.1d).

Now we prove the inequalities. The inequality $v_{\mathrm{IP}} \leqslant v_{\mathrm{his}}^*$ holds because $\boldsymbol{\delta}^{\mathrm{IP}} \in \Delta_{\mathrm{his}}$. It remains to show that $z_{\mathrm{IP}} \leqslant v_{\mathrm{IP}}$. We do it using a backward induction. Let $(\boldsymbol{\tau}^{*m}, \boldsymbol{\delta}^{*m})_{m\in[M]}$ be an optimal solution of MILP (5.1). We denote by $\mathrm{P}_t(\mathbf{h}_t)$ the feasible set of the optimization problem solved in $\mathrm{Act}_T^{\mathrm{IP},t}(\mathbf{h}_t)$, for every t in $[T]$. We consider the following induction hypothesis at time t:

$$\max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]} \in \mathrm{P}_t(\mathbf{h}_t)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t'=t}^{T} r^m(S_t^m, A_t^m, S_{t+1}^m) | H_t^m = h_t^m\right] \leqslant \mathbb{E}_{\boldsymbol{\delta}^{\mathrm{IP}}}\left[\sum_{m=1}^{M}\sum_{t'=t}^{T} r^m(S_t^m, A_t^m, S_{t+1}^m) | \mathbf{H}_t = \mathbf{h}_t\right]$$

If $t = T$, then consider left-hand side is exactly equal to the right-hand side.

$$\begin{aligned}&\max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]} \in \mathrm{P}_T(\mathbf{h}_T)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[r(S_T^m, A_T^m, S_{T+1}^m) | H_T^m = h_T^m\right] \\ =& \max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]} \in \mathrm{P}_T(\mathbf{h}_T)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[r(S_T^m, A_T^m, S_{T+1}^m) | S_T^m \sim p^m(\cdot|h_T^m)\right] = \mathbb{E}_{\boldsymbol{\delta}^{m,\mathrm{IP}}}\left[\sum_{m=1}^{M} r^m(S_T^m, A_T^m, S_{T+1}^m) | \mathbf{H}_T = \mathbf{h}_T\right]\end{aligned}$$

The first equality comes from the fact that the belief state is a sufficient statistic of the history. It proves the induction hypothesis for $t = T$.

Suppose that the induction hypothesis holds from $t+1$. We compute the term in t:

$$
\begin{aligned}
&\max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}\in \mathrm{P}_t(\mathbf{h}_t)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t'=t}^{T} r^m(S_{t'}^m, A_{t'}^m, S_{t'+1}^m)|H_t^m = h_t^m\right] \\
&= \max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}\in \mathrm{P}_t(\mathbf{h}_t)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^{t,m}}\left[r^m(S_t^m, A_t^m, S_{t+1}^m)|H_t^m = h_t^m\right] \\
&+ \sum_{m=1}^{M} \sum_{a_t^m, o_{t+1}^m} \Bigg(\mathbb{P}_{\boldsymbol{\delta}t,m}\left(O_{t+1}^m = o_{t+1}^m, A_t^m = a_t^m | H_t^m = h_t^m\right) \\
&\overbrace{\times \mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t'=t+1}^{T} r^m(S_{t'}^m, A_{t'}^m, S_{t'+1}^m)|\underbrace{H_t^m = h_t^m, A_t^m = a_t^m, O_{t+1}^m = o_{t+1}^m}_{H_{t+1}^m = h_{t+1}^m}\right]}^{\text{does not depend on } \boldsymbol{\delta}^{t,m}}\Bigg) \\
&\leqslant \mathbb{E}_{\boldsymbol{\delta}^{t,\mathrm{IP}}}\left[\sum_{m=1}^{M} r^m(S_t^m, A_t^m, S_{t+1}^m)|\mathbf{H}_t = \mathbf{h}_t\right] + \Bigg(\sum_{\mathbf{a}_t,\mathbf{o}_{t+1}} \mathbb{P}_{\boldsymbol{\delta}t,\mathrm{IP}}(\mathbf{O}_{t+1} = \mathbf{o}_{t+1}, \mathbf{A}_t = \mathbf{a}_t | \mathbf{H}_t = \mathbf{h}_t) \\
&\overbrace{\times \max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)\in \mathrm{P}_{t+1}(\mathbf{h}_{t+1})} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t'=t+1}^{T} r^m(S_{t'}^m, A_{t'}^m, S_{t'+1}^m)|H_{t+1}^m = h_{t+1}^m\right]}^{\text{induction hypothesis}}\Bigg) \\
&\leqslant \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^{m,\mathrm{IP}}}\left[r^m(S_t^m, A_t^m, S_{t+1}^m)|H_t^m = h_t^m\right]
\end{aligned}
$$

The first inequality above comes from the fact that there exists an optimal solution where $\boldsymbol{\delta}^{t,\mathrm{IP}}$ is the policy at time t by definition of $\boldsymbol{\delta}^{\mathrm{IP}}$ and by decomposing the maximum operator in the sum of the second term. This latter operation can be done since $\mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t'=t+1}^{T} r^m(S_{t'}^m, A_{t'}^m, S_{t'+1}^m)|H_{t+1}^m = h_{t+1}^m\right]$ does not depend on the policy $\boldsymbol{\delta}^{t',m}$ for $t' < t+1$. It proves the backward induction. Finally, given an optimal feasible solution $(\boldsymbol{\tau}*^m, \boldsymbol{\delta}*^m)_{m\in[M]}$ of MILP (5.1), we get that:

$$
\begin{aligned}
z_{\mathrm{IP}} = \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}*^m}\left[\sum_{t=1} r^m(S_t^m, A_t^m, S_{t+1}^m)\right] &= \mathbb{E}\left[\mathbb{E}_{\boldsymbol{\delta}*^m}\left[\sum_{m=1}^{M}\sum_{t=1}^{T} r^m(S_t^m, A_t^m, S_{t+1}^m)|O_1^m = o_1^m\right]\right] \\
&\leqslant \mathbb{E}\left[\max_{(\boldsymbol{\tau}^m,\boldsymbol{\delta}^m)_{m\in[M]}\in \mathrm{P}_1(\mathbf{h}_1)} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^m}\left[\sum_{t=1} r^m(S_t^m, A_t^m, S_{t+1}^m)|O_1^m = o_1^m\right]\right] \\
&\leqslant \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{\delta}^{m,\mathrm{IP}}}\left[\sum_{t=1} r^m(S_t^m, A_t^m, S_{t+1}^m)\right] = v_{\mathrm{IP}}
\end{aligned}
$$

The first inequality comes from the inversion of the maximum operator and the expectation. It achieves the proof. □

5.6 Rolling horizon heuristic

When the horizon T is long it is computationally interesting to embed the implicit policy (5.14) in a rolling horizon heuristic, which consists in repeatedly solving an optimization problem with a shorter horizon at each time step and to take action at the current time.

Indeed, in theory, computing the implicit feasible policy (5.14) requires to compute $\mathrm{Act}_T^t(\mathbf{h})$ for every history $\mathbf{h} \in \mathcal{X}_H^t$ and every time t in $[T]$. It leads to two computational difficulties. First, since the size of $\mathcal{X}_H^t$ is exponential, computing the feasible policy becomes quickly prohibitive. Second, if the finite horizon T is too large, then solving MILP (5.1) in Step 4 becomes intractable.

In practice, at each time t the decision maker receives an observation $\mathbf{o}_t$ in $\mathcal{X}_O$ and takes an action $\mathbf{a}_t$ in $\mathcal{X}_A$. Hence, it only requires to compute $\mathrm{Act}_T^t(\mathbf{h}_t)$ where $\mathbf{h}_t = (\mathbf{h}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)$, which addresses the first issue. To address the second issue, we compute MILP (5.1) using a smaller rolling horizon $T_\mathrm{r} < T$. The following algorithm shows how we use the feasible policy in practice:

Algorithm 3 Rolling horizon heuristic.

```
1: Input: $T$, $T_\mathrm{r}$, $\big((\mathcal{X}_S^m, \mathcal{X}_O^m, \mathcal{X}_A^m, \mathfrak{p}^m, \mathbf{r}^m, \mathbf{D}^m)_{m\in[M]}, \mathbf{b}\big)$
2: for $t = 1, \dots, T$ do
3:     Receive observation $\mathbf{o}_t$
4:     Take action $\mathrm{Act}_{t+T_\mathrm{r}}^t(\mathbf{h}_t)$
5: end for
```

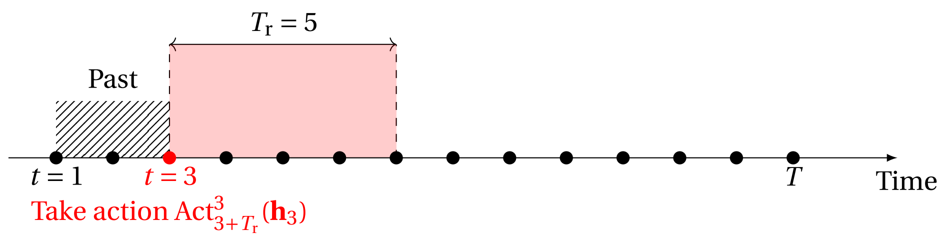
Figure 5.2 illustrates two consecutive iterations of Algorithm 3 with a rolling horizon $T_\mathrm{r} = 5$. This type of rolling horizon heuristic is commonly used for multistage optimization problems in Operations Research [92, 127, 128, 136]. Numerical experiments in Chapter 10 show the efficiency of the implicit policy (5.14).

5.7 Numerical experiments

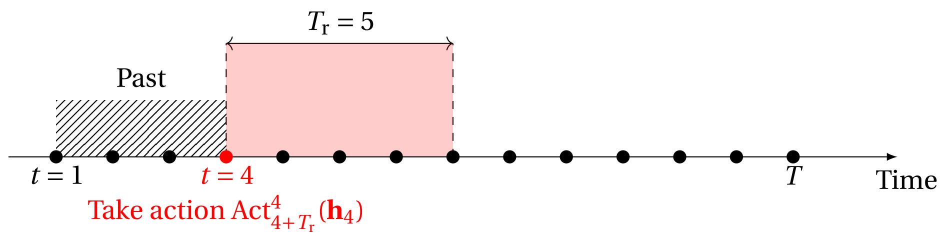
In this section, we provide numerical experiments on the partially observable multi-armed bandits problem introduced in Example 1. All mathematical programs have been written in `Julia` [18] with the `JuMP` [41] interface and solved using `Gurobi` 9.0. [52] with the default settings. Experiments have been run on a server with 192Gb of RAM and 32 cores at 3.30GHz. We show the quality of the approximation (5.1) by comparing the values of z_{LB}, z_{IP}, z_{UB}, z_{LR}, $z_{\mathrm{R^c}}$ and z_{R}. The results on a predictive maintenance problem with capacity constraints are reported in Part III.

Instances. We consider instances where the state space and observation space of each bandit have the same cardinality n, i.e., $n := |\mathcal{X}_S^m| = |\mathcal{X}_O^m|$ for any m in $[M]$. The resulting system's state space and system's observation space have the size $|\mathcal{X}_S| = |\mathcal{X}_O| = n^M$. In each bandit state space, the states and observations are numbered from 1 to n, i.e., $\mathcal{X}_S^m = \mathcal{X}_O^m = \{1, \dots, n\}$. Like Bertsimas and Mišić [15], we generate different set of 10 instances: regular (REG.SAR), restless (RSTLS.SAR and RSTLS.SBR) or deterministic (RSTLS.DET.SBR) multi-armed bandits. For each set of instance, the emission probability vector $(p^m(o|s))_{o\in\mathcal{X}_O^m, s\in\mathcal{X}_S^m}$ is uniformly drawn from $[0,1]$ and renormalized. Each set contains 10 instances. The sets of instances are generated as follows.

- REG.SAR consists of regular partially observable multi-armed bandits. The reward func-



(a) Iteration $t = 3$ of Algorithm (3)



(b) Iteration $t = 4$ of the Algorithm 3

Figure 5.2 – Scheme of the evaluation of our implicit policy (5.14) in Algorithm 3 at time $t = 3$ and $t = 4$. The decision maker observes $\mathbf{h}_3$ and takes action $\mathrm{Act}^3_8(\mathbf{h}_3)$. Then, the decision maker observes $\mathbf{h}_4$ and takes action $\mathrm{Act}^4_9(\mathbf{h}_4)$. The black points indicate the time steps and the red point corresponds to the time when the decision is taken. The black hatched lines represent the past at the current time (red). The red square indicates the horizon taken into account in the optimization problem.

tion is defined by $r^m(s,1,s') := (10/n)\cdot s$ and $r^m(s,0,s') := 0$ for every state $s, s' \in \mathcal{X}^m_S$ and every arm $m \in [M]$. Each active transition probability vector $(p^m(s'|s,1))_{s,s' \in \mathcal{X}^m_S}$ is drawn uniformly from $[0,1]$ and renormalized, for every arm $m \in [M]$. Each passive arm m stays in the same state, i.e., $p^m(s'|s,0) = \mathbb{1}_s(s')$ for every $s, s' \in \mathcal{X}^m_S$.

- RSTLS.SAR consists of restless partially observable multi-armed bandits. The reward function is the same as REG.SAR. Each active and passive transition probability vector $(p^m(s'|s,a))_{\substack{s,s' \in \mathcal{X}^m_S \\ a \in \{0,1\}}}$ is drawn uniformly from $[0,1]$ and renormalized, for every arm $m \in [M]$.
- RSTLS.SBR consists of restless partially observable multi-armed bandits. The reward function is defined by $r^m(s,1,s') := (10/n)\cdot s$ and $r^m(s,0,s') := (1/M)\cdot(10/n)\cdot s$ for every state $s, s' \in \mathcal{X}^m_S$ and every arm $m \in [M]$. The transition probability is randomly drawn as RSTLS.SAR.
- RSTLS.DET.SBR consists of restless partially observable multi-armed bandits. The reward function is the same as RSTLS.SBR. Each active and passive transition probability vector $(p^m(s'|s,a))_{\substack{s,s' \in \mathcal{X}^m_S \\ a \in \{0,1\}}}$ is randomly drawn and deterministic, for every arm $m \in [M]$.

Bertsimas and Mišić [15] explained the benefits of using such transition probabilities and reward functions. We generate small-scale instances with $M \in \{2,3\}$ arms and $n = 4$ states, and medium-scale instances with $M = 5$ arms and $n = 4$ states.

Metrics. For each instance, we compute the value z_{IP}, the lower bound z_{LB} and the upper bounds z_{UB}, z_{LR}, $z_{\mathrm{R^c}}$ and z_{R}. Given an instance, we define the relative gaps with the largest upper bound z_{R}: $\mathrm{g_{LB}} = \frac{z_{\mathrm{R}} - z^{\mathrm{LB}}}{z_{\mathrm{R^c}}}$, $\mathrm{g_{IP}} = \frac{z_{\mathrm{R}} - z_{\mathrm{IP}}}{z_{\mathrm{R^c}}}$, $\mathrm{g_{UB}} = \frac{z_{\mathrm{R^c}} - z_{\mathrm{UB}}}{z_{\mathrm{R^c}}}$ and $\mathrm{g_{LR}} = \frac{z_{\mathrm{R^c}} - z_{\mathrm{LR}}}{z_{\mathrm{R^c}}}$. Then, we define respectively the metrics $G_{\mathrm{mean}}(\mathrm{g})$, $G_{95}(\mathrm{g})$ and $G_{\max}(\mathrm{g})$ as the mean, the 95-th percentile and the maximum over a set of instances, for each gap g in $\{\mathrm{g_{LB}}, \mathrm{g_{IP}}, \mathrm{g_{UB}}, \mathrm{g_{LR}}\}$. In general, the lower the values of the metrics, the closer the bound is to the upper bound $z_{\mathrm{R^c}}$. In particular, thanks to Theorem 5.4 and Proposition 5.6the metrics $\mathrm{g_{LB}}$ and $\mathrm{g_{LR}}$ tell how close are the values of z_{IP} and v^*_{ml}. Since the computation of z_{UB} becomes quickly difficult when the sizes of the instance increase, we only compute the values of $\mathrm{g_{UB}}$ on small instances. We also report the mean computation time over the

Table 5.2 summarizes the results averaged over the 10 instances of each set. For all the mathematical programs, we set the computation time limit to 3600 seconds. If the resolution has not terminated before this time limit, then we keep the best feasible solution obtained at the end of the resolution. It explains why for some instances we obtain a smaller gap with lower bound (5.3) than with MILP (5.1). The Lagrangian relaxation value z_{LR} is computed using a column generation approach.

One can observe in Table 5.2 that for almost a large part of the instances, the values of z_{LB}, z_{IP}, z_{UB}, and z_{LR} are close in general. It shows that our formulations have optimal values that are close to the optimal value v^*_{ml} of $(\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}})$. In addition, the best bound obtained on the value of z_{IP} is very close to the value of the lower bound z_{LB}. Thanks to Theorem 5.3, it means that most of the multi-armed bandit instances admit optimal policies that are "decomposable" (see Section 5.4).

5.7.1 Simulations of the implicit policy

The aim of this section is to show how the value returned by matheuristic 3 is close to the optimal value v^*_{his}, and that policy $\boldsymbol{\delta}^{\mathrm{IP}}$ can be computed in a reasonable amount of time on large-scale instances of a practical problem. We evaluate the performances of the history-dependent policy (5.14) by running Algorithm 3 on a maintenance problem taken from the literature. Like Walraven and Spaan [157, Section 5.2], we consider a road authority that performs maintenance on M bridges, each of them evolving independently over a finite horizon H. Each bridge is modeled as a POMDP [44], and the authority must chooses at most K bridges to maintain at each decision time. As mentioned in Section 3.3, this problem can be modeled as a weakly coupled POMDP.

Instances Like Walraven and Spaan [157], we build our instances of weakly coupled POMDP from the `bridge-repair` instance of Ellis et al. [44] in which the decision maker has to perform maintenance on a bridge. In our problem, there are only two actions available on each bridge: either structural repair or keep. For each bridge m, the sizes of state space, observation space and action space are respectively $|\mathcal{X}^m_S| = 5$, $|\mathcal{X}^m_O| = 5$ and $|\mathcal{X}^m_A| = 2$. Each bridge starts almost surely in its most healthy state. We add noises to the transition probabilities and emission probabilities of the bridges to ensure that they have slightly different parameters $\mathfrak{p}^m$ for all m in $[M]$. For every bridge m in $[M]$, we set $C^m_F = 1000$ and $C^m_R = 100$. The bridges are inspected every months and evolve until the horizon of $H = 24$ months. One instance consists in the value

Instance set	T	g	$M=2$				$M=3$				$M=5$			
			$G_{\text{mean}}(\text{g})$	$G_{95}(\text{g})$	$G_{\max}(\text{g})$	Time(s)	$G_{\text{mean}}(\text{g})$	$G_{95}(\text{g})$	$G_{\max}(\text{g})$	Time(s)	$G_{\text{mean}}(\text{g})$	$G_{95}(\text{g})$	$G_{\max}(\text{g})$	Time(s)
REG.SAR	2	g_{LB}	9.91	15.13	15.62	0.03	15.72	21.51	22.87	0.08	17.42	25.08	26.53	0.21
		g_{IP}	9.91	15.13	15.62	0.14	15.72	21.51	22.87	0.64	17.42	25.08	26.53	2.57
		g_{UB}	7.54	10.54	10.70	0.16	11.18	17.89	19.40	0.36	–	–	–	–
		g_{LR}	7.02	10.21	10.33	9.55	10.86	16.97	18.73	14.08	13.00	18.41	18.64	15.52
	5	g_{LB}	6.06	9.23	9.57	0.39	10.34	13.01	13.26	1.32	14.51	18.45	18.48	3.43
		g_{IP}	6.06	9.23	9.57	17.14	10.34	13.01	13.26	1525.63	17.10	27.03	27.04	2907.11
		g_{UB}	4.59	6.07	6.19	1260.58	7.18	9.89	9.99	3247.98	–	–	–	–
		g_{LR}	4.04	5.89	5.95	43.00	6.79	9.36	9.42	54.66	10.46	13.15	13.61	51.17
	10	g_{LB}	3.36	5.45	5.85	2.86	6.27	8.91	9.15	39.76	8.04	11.14	11.55	349.00
		g_{IP}	3.38	5.55	6.04	1283.79	10.08	18.86	19.93	3205.30	15.53	23.63	24.08	> 3600
		g_{UB}	5.08	9.41	9.51	3248.01	–	–	–	–	–	–	–	–
		g_{LR}	2.13	3.27	3.46	946.01	3.90	5.74	5.84	1536.79	5.98	7.52	7.67	661.04
RSTLS.SAR	2	g_{LB}	12.73	16.84	17.04	0.03	17.96	22.67	22.77	0.07	15.13	16.83	17.03	0.17
		g_{IP}	12.73	16.84	17.04	0.14	17.96	22.67	22.77	0.62	15.13	16.83	17.03	2.28
		g_{UB}	8.33	13.09	15.56	0.07	12.01	18.18	18.93	0.06	–	–	–	–
		g_{LR}	8.16	12.71	14.99	9.22	11.99	18.15	18.91	14.44	9.94	11.20	11.28	15.79
	5	g_{LB}	10.77	13.61	14.22	0.44	14.54	18.64	18.86	1.40	13.85	16.46	17.00	3.91
		g_{IP}	10.77	13.61	14.22	12.50	14.54	18.64	18.86	358.36	18.31	21.51	21.66	3030.09
		g_{UB}	6.55	8.51	8.73	171.29	8.14	12.43	13.86	1716.52	–	–	–	–
		g_{LR}	6.32	8.16	8.35	41.99	7.92	12.01	13.67	36.89	7.92	9.86	10.19	59.57
	10	g_{LB}	10.39	13.73	14.32	3.20	13.18	16.36	16.83	29.58	14.55	17.85	18.18	312.76
		g_{IP}	10.86	15.35	17.28	1896.65	14.42	17.97	18.07	> 3600	18.83	26.05	27.44	> 3600
		g_{UB}	5.99	8.44	9.28	> 3600	–	–	–	–	–	–	–	–
		g_{LR}	5.44	7.29	7.83	> 3600	5.74	6.93	7.05	1669.80	7.34	9.42	9.68	> 3600
RSTLS.SBR	2	g_{LB}	6.14	8.70	9.06	0.02	8.50	11.51	11.67	0.03	9.07	10.98	11.54	0.07
		g_{IP}	6.14	8.70	9.06	0.04	8.50	11.51	11.67	0.14	9.07	10.98	11.54	0.72
		g_{UB}	3.54	5.63	6.00	0.02	5.39	7.60	7.75	0.02	–	–	–	–
		g_{LR}	3.47	5.49	5.75	6.72	5.32	7.56	7.71	7.74	6.39	7.63	7.99	7.90
	5	g_{LB}	4.65	7.58	7.84	0.18	6.77	9.13	9.24	0.66	8.61	10.38	10.51	1.72
		g_{IP}	4.65	7.58	7.84	4.27	6.77	9.13	9.24	311.29	10.84	14.81	16.37	3291.89
		g_{UB}	2.61	4.80	5.78	18.55	3.61	5.26	5.43	500.43	–	–	–	–
		g_{LR}	2.33	4.16	4.85	14.43	3.43	5.14	5.27	21.78	5.10	5.89	5.91	19.59
	10	g_{LB}	4.27	7.82	8.52	1.29	6.00	8.86	9.20	16.71	7.99	10.75	11.28	127.66
		g_{IP}	4.37	7.90	8.52	890.65	9.87	15.37	15.70	3265.16	15.84	17.97	18.01	> 3600
		g_{UB}	2.31	4.96	6.48	1836.06	–	–	–	–	–	–	–	–
		g_{LR}	1.86	3.42	3.86	3168.63	2.77	4.69	4.76	3494.95	4.73	5.65	5.80	1443.62
RSTLS.DET.SBR	2	g_{LB}	6.08	11.58	13.78	0.02	8.52	13.94	14.43	0.03	9.74	13.54	13.57	0.11
		g_{IP}	6.08	11.58	13.78	0.05	8.52	13.94	14.43	0.26	9.74	13.54	13.57	1.76
		g_{UB}	4.70	8.33	9.45	0.02	6.54	9.24	9.59	0.03	–	–	–	–
		g_{LR}	4.67	8.31	9.42	12.44	6.35	9.22	9.56	14.00	7.43	10.17	10.60	13.76
	5	g_{LB}	2.99	7.26	8.87	0.12	6.15	10.06	10.98	0.68	5.88	8.21	8.95	1.76
		g_{IP}	2.99	7.26	8.87	0.17	6.24	10.06	10.98	6.47	5.88	8.21	8.95	32.56
		g_{UB}	2.72	6.86	8.66	2.97	5.01	8.11	9.52	77.43	–	–	–	–
		g_{LR}	2.45	6.31	8.15	17.40	4.58	7.26	8.62	21.58	4.95	7.41	7.58	26.48
	10	g_{LB}	2.39	6.94	7.50	0.59	4.29	8.36	10.01	3.90	4.70	7.54	8.25	43.95
		g_{IP}	2.39	6.94	7.50	4.79	4.29	8.36	10.01	514.32	6.38	13.10	13.87	1148.11
		g_{UB}	2.23	7.04	7.60	816.32	–	–	–	–	–	–	–	–
		g_{LR}	1.89	6.12	7.22	76.55	3.10	5.92	6.70	197.00	3.98	6.68	7.02	116.91

Table 5.2 – The values of $G_{\text{mean}}(\text{g})$, $G_{95}(\text{g})$, and $G_{\max}(\text{g})$ obtained on the small-scale and medium-scale instances with $M \in \{2,3,5\}$, $n = 4$ and solved with different finite horizon $T \in \{2,5,10\}$.

of the tuple $(M, K, (\mathfrak{p}^m)_{m\in[M]})$. We build an instance as follows: first we choose a value of M in $\{3,4,5,10,15,20\}$, second we build the probabilities $\mathfrak{p}^m$ by adding a random real in $[0,0.1]$ to each non-zero value of the probabilities of Ellis et al. [44], and finally we choose $K = \max(\lfloor\omega \times M\rfloor, 1)$, where ω is a scalar belonging $\{0.2, 0.4, 0.6, 0.8\}$ (when $M = 3$, then K belongs to $\{1,2\}$). The range of values of K is chosen in such a way that it goes from highly restrictive constraints (smallest values of ω) to more flexible constraints (largest values of ω), with respect to the value of M. When $K \geqslant M$, the decision maker can consider the subproblems separately, which is much easier. We enforce K to be non smaller than 1 because if $K = 0$, the authority cannot maintain the bridges.

We evaluate the performances of matheuristic 3 for rolling horizon T_{r} in $\{2,5\}$. For each instance $(M, K, (\mathfrak{p}^m)_{m\in[M]})$, we perform 10^3 runs of matheuristic 3. We compute the average total cost $|\nu_{\mathrm{IP}}|$, the average number of failures f_{IP} over the 10^3 simulations. We compare ν_{IP} with the upper bound $z_{\mathrm{R^c}}$ and the Lagrangian bound z_{LR} by evaluating the average gap $\mathrm{G}_{\mathrm{IP}}^{\mathrm{R^c}} = \frac{z_{\mathrm{R^c}} - \nu_{\mathrm{IP}}}{|z_{\mathrm{R^c}}|}$ and $\mathrm{G}_{\mathrm{IP}}^{\mathrm{LR}} = \frac{z_{\mathrm{LR}} - \nu_{\mathrm{IP}}}{|z_{\mathrm{LR}}|}$. Thanks to Theorem 5.2, the value of $G_{\mathrm{IP}}^{\mathrm{R^c}}$ indicates how far is ν_{IP} from v_{his}^* because $\nu_{\mathrm{IP}} \leqslant v_{\mathrm{his}}^* \leqslant z_{\mathrm{R^c}}$. The lower the value of $G_{\mathrm{IP}}^{\mathrm{R^c}}$, the better is the performance of policy $\boldsymbol{\delta}^{\mathrm{IP}}$. In addition, for each simulation, we compute the average computation time in seconds of the underlying formulation over all steps of the simulation. We then consider the average value over all the N simulations. For the quantities $|\nu_{\mathrm{IP}}|$ and f_{IP} we also report the standard deviations over all simulations.

Table 5.3 summarizes the results. For all the mathematical programs, we set the computation time limit to 3600 seconds and a final gap tolerance (`MIPGap` parameter in `Gurobi`) of 1%, which is enough for the use of our matheuristic. If the resolution has not terminated before this time limit, then we keep the best feasible solution at the end of the resolution.

One may observe that for all instances, the matheuristic involving our MILP (5.1) delivers promising results even in the most challenging instance ($M = 20$). In particular, the values of $\mathrm{G}_{\mathrm{IP}}^{\mathrm{R^c}}$ show that the policy $\boldsymbol{\delta}^{\mathrm{IP}}$ gives an optimality gap (in the set of history-dependent policies) of at most 10% on the large-scale instance, which is satisfying regarding the complexity of the optimization problem. In Table 5.3, the negative values of $\mathrm{G}_{\mathrm{IP}}^{\mathrm{R^c}}$ result from error approximations due to the Monte-Carlo simulations. It can also be noted that the gap $\mathrm{G}_{\mathrm{IP}}^{\mathrm{LR}}$ is takes negative values for some instances, which shows that ν_{IP} can take larger values than the Lagrangian relaxation for some instances. It highlights the benefit of using the belief state updates in the definition of $\boldsymbol{\delta}^{\mathrm{IP}}$. In addition, even for the largest instances ($M = 15$ or $M = 20$) and for $T = 5$, the average time per action of $\mathrm{Act}_{t+T_{\mathrm{r}}}^{\mathrm{IP},t}(\mathbf{h}_t)$ is on the order of 1.0 second; this amount of time is still feasible even if the 24 decision times are close together.

5.8 Bibliographical remarks

$(\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}})$ lies in the broad class of multi-stage stochastic optimization problems with high-dimensional state spaces. As mentioned in Section 3.5, $(\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}})$ is a weakly coupled dynamic program where the decision maker has only access to partial observations of the system.

When the decision maker has access to the system's state, $(\mathrm{P}_{\mathrm{ml}}^{\mathrm{wc}})$ becomes a weakly coupled dynamic program. It is well known that such a problem can be solved using dynamic programming approach by writing the Bellman's equation. Since the size of the state space and

M	ω	T_r	$\lvert v_{\mathrm{IP}}\rvert$	Std. err.	$\mathrm{f_{IP}}$	Std. err.	$\mathrm{G_{IP}^{LR}}$	$\mathrm{G_{IP}^{R^c}}$	Time
			$(\times 10^3)$	$(\times 10^3)$			(%)	(%)	(s)
3	0.2	2	5.71	2.32	4.6	2.2	11.60	14.09	0.004
		5	5.45	2.16	4.0	2.1	**6.51**	**8.88**	**0.055**
	0.4	2	5.71	2.32	4.6	2.2	11.60	14.09	0.005
		5	5.45	2.16	4.0	2.1	**6.51**	**8.88**	**0.062**
	0.6	2	5.71	2.32	4.6	2.2	11.60	14.09	0.006
		5	5.45	2.16	4.0	2.1	**6.51**	**8.88**	**0.064**
	0.8	2	5.17	1.90	4.0	1.8	0.95	3.21	0.005
		5	5.11	1.86	3.5	1.8	**-0.12**	**2.12**	**0.056**
4	0.2	2	8.81	2.87	6.5	2.9	-1.77	13.79	0.007
		5	8.64	2.92	6.3	2.9	**-3.64**	**11.62**	**0.116**
	0.4	2	8.81	2.87	6.5	2.9	-1.77	13.79	0.007
		5	8.64	2.92	6.3	2.9	**-3.64**	**11.62**	**0.118**
	0.6	2	7.70	1.96	4.3	1.9	-0.94	0.46	0.006
		5	7.62	1.89	4.0	1.9	**-2.05**	**-0.66**	**0.067**
	0.8	2	7.70	1.91	4.1	1.8	-0.94	0.45	0.006
		5	7.61	1.81	3.6	1.7	**-2.16**	**-0.79**	**0.060**
5	0.2	2	13.78	4.77	11.5	4.8	-41.47	30.24	0.008
		5	13.38	4.58	11.1	4.6	**-43.18**	**26.44**	**0.250**
	0.4	2	10.43	2.50	6.4	2.5	0.76	2.13	0.007
		5	10.27	2.39	5.7	2.4	**-0.77**	**0.58**	**0.124**
	0.6	2	10.26	2.31	5.7	2.2	0.00	0.80	0.006
		5	10.22	2.26	5.2	2.2	**-0.44**	**0.35**	**0.071**
	0.8	2	10.24	2.28	5.6	2.2	-0.27	0.58	0.006
		5	10.20	2.22	5.1	2.2	**-0.65**	**0.20**	**0.070**
10	0.2	2	22.63	5.45	18.0	5.5	-34.20	17.71	0.012
		5	22.06	5.24	17.5	5.2	**-35.85**	**14.74**	**0.384**
	0.4	2	19.19	3.38	11.5	3.3	1.32	3.07	0.012
		5	18.91	3.26	10.6	3.2	**-0.16**	**1.57**	**0.196**
	0.6	2	19.10	3.28	10.8	3.1	0.92	2.60	0.011
		5	18.81	3.06	9.7	2.9	**-0.62**	**1.03**	**0.138**
	0.8	2	19.09	3.27	10.8	3.1	0.86	2.53	0.011
		5	18.82	3.08	9.6	2.9	**-0.56**	**1.09**	**0.137**
15	0.2	2	31.54	6.03	25.0	6.0	-22.56	12.73	0.017
		5	30.89	5.88	24.0	5.9	**-24.14**	**10.43**	**0.591**
	0.4	2	28.18	4.22	19.1	4.1	0.45	1.93	0.016
		5	27.67	3.97	16.8	3.9	**-1.39**	**0.06**	**0.232**
	0.6	2	28.12	4.16	18.8	4.0	0.10	1.70	0.016
		5	27.67	3.84	16.3	3.7	**-1.51**	**0.05**	**0.225**
	0.8	2	28.12	4.16	18.8	4.0	0.11	1.71	0.015
		5	27.65	3.86	16.2	3.7	**-1.57**	**-0.01**	**0.226**
20	0.2	2	45.06	7.01	35.9	7.1	-20.37	8.67	0.022
		5	44.28	6.90	35.1	6.9	**-21.74**	**6.80**	**0.660**
	0.4	2	41.18	4.72	23.0	4.7	0.35	1.50	0.020
		5	40.83	4.66	22.7	4.7	**-0.49**	**0.66**	**0.469**
	0.6	2	40.96	4.25	18.4	4.1	0.32	1.22	0.020
		5	40.72	4.19	17.9	4.0	**-0.28**	**0.62**	**0.316**
	0.8	2	40.96	4.24	18.3	4.0	0.29	1.21	0.019
		5	40.76	4.09	17.8	3.9	**-0.19**	**0.73**	**0.313**

Table 5.3 – Performances of the matheuritic on different rolling horizon $T_r \in \{2,5\}$: Numerical values of $\lvert v_{\mathrm{IP}}\rvert$, $\mathrm{f_{IP}}$ (and their corresponding standard errors), $\mathrm{G_{IP}^{LR}}$ and $\mathrm{G_{IP}^{R^c}}$ obtained on an instance (M,ω) with $M \in \{3,4,5,10,15,20\}$ and $\omega \in \{0.2,0.4,0.6,0.8\}$. The values written in bold indicate the best performances of policy $\boldsymbol{\delta}^{\mathrm{IP}}$ regarding optimality and scalability (computation time).

action space are exponential in the number of components of the system, the main challenge of solving weakly coupled dynamic programs is the curse of dimensionality as mentioned in Chapter 3, which makes, in general, the exact dynamic programming approaches intractable. To address this challenge, several approximate dynamic programming methods have been proposed [14, 46, 105, 123]. These approaches are mainly based on an approximation of the value functions. Other approaches to solve weakly coupled dynamic program consist in deriving heuristic policies from relaxations. These relaxations are of two types: Lagrangian relaxations of the linking constraints in the definition (3.2) of $\mathcal{X}_A$ [2, 55, 165], and relaxations of the *non-anticipativity constraints* [20], which assume that the decision maker has access to the future outcomes. Recently, Bertsimas and Mišić [15] gives the tightest upper bound for *decomposable MDPs* with discounted rewards over infinite horizon, which is a more general case where the action space does not necessary decompose along the component action spaces (see remark 2). Their approach and our approach are both based on the linear formulation in terms of moments for Markov Decision Process (4.13).

When the decision maker has access to the system's state, the predictive maintenance problem with capacity constraints corresponds to the restless multi-armed bandit problem in finite horizon, which is a special case of weakly coupled dynamic program. Such problem is known to be PSPACE-hard [114]. The usual heuristic policies to solve restless multi-armed bandit problem are the *index policies* introduced by Gittins [50]. An index policy consists in computing an index for each arm separately and selecting the arms with the highest indices. It enables to decompose the computation. Whittle index policy [162] is the most used policy, which is practically efficient in various applications. Other index policies based on polyhedral approaches have been proposed [16, 17, 111].

In fact, the predictive maintenance problem with capacity constraints is at least as difficult as the restless multi-armed bandit problem. In addition to the curse of dimensionality, the fact that the system is partially observable represents a second challenge. To address this challenge, new index policies have been proposed [1, 67, 104]. Their approach are based on the fact that restless partially observable multi-armed bandit is restless multi-armed bandit where the state space of each component is its belief state space. However, the results proposed hold when the state spaces and observation spaces contain at most two elements.

Part II

Integer programming for influence diagrams

6 Maximum Expected Utility in influence diagrams

This chapter introduces Influence Diagrams, which form a flexible tool that enables to model discrete stochastic optimization problems, including Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs) as standard examples. More precisely, given random variables considered as vertices of an directed acyclic graph, a directed probabilistic graphical model defines a joint distribution via the conditional distributions of vertices given their parents. In influence diagrams, the random variables are represented by the set of vertices of an acyclic directed graph that is partitioned into three types of vertices: chance, decision and utility vertices. It is assumed that the probability distributions of the chance and utility vertices conditionally to their parents are known. The decision maker chooses the probability distribution of the decision vertices conditionally to their parents in order to maximize the expected utility. Through examples, we show the modeling power of the influence diagrams and we describe the main results and solution algorithms we propose to solve such a maximization problem. These results have been partially published in Parmentier et al. [117]. Several results have been added in this dissertation after the publication of the paper.

Chapter 6 is organized as follows:

- Section 6.1 recalls the definition of directed graphical model and the maximum expected utility problem in influence diagrams. We also provide examples of discrete stochastic optimization problem that can be modeled using influence diagrams.
- Section 6.2 introduces the key notions required to read the main results of Part II.
- Section 6.3 states the main results of Part II. First, we present an mixed-integer linear formulation for solving exactly the maximum expected utility problem. Second, we introduce valid inequalities for our formulation, which lead to a computationally efficient algorithm. Third, we show that the linear relaxation of our integer formulation yields optimal integer solutions for instances that can be solved by the "single policy update," the standard algorithm for addressing influence diagrams.

6.1 The Influence Diagrams

Now we recall the framework of influence diagrams (more details can be found in Koller and Friedman [75, Chapter 23]). We choose to use the terminology and notation of the probabilistic graphical model literature [156] instead of that of graph theory or combinatorial optimization.

Let $G = (V, A)$ be a directed graph with a set of vertex V and a set of oriented arcs A. A *parent* (resp. *child*) of a vertex v is a vertex u such that (u, v) (resp. (v, u)) belongs to A; we denote by $\mathrm{pa}(v)$ the set of parents vertices (resp. $\mathrm{ch}(v)$ the set of children vertices). The *family* of v, denoted by $\mathrm{fa}(v)$, is the set $\{v\} \cup \mathrm{pa}(v)$.

6.1.1 The framework of parametrized influence diagram

Let $G = (V, A)$ be an acyclic directed graph, and, for each vertex v in V, let X_v be a random variable taking value in a finite state space $\mathcal{X}_v$. For any $C \subseteq V$, let X_C denote $(X_v)_{v \in C}$ and $\mathcal{X}_C$ be the Cartesian product $\mathcal{X}_C = \prod_{v \in C} \mathcal{X}_v$. A *directed probabilistic graphical model* (or more concisely *directed graphical model*) is an acyclic directed graph G and the collection of probability distributions $\mathbb{P}$ of the random vector X_V such that there exists a collection of conditional probability distributions $\{p_{v|\mathrm{pa}(v)}\}_{v \in V}$ satisfying

$$\mathbb{P}(X_V = x_V) = \prod_{v \in V} p_{v|\mathrm{pa}(v)}(x_v | x_{\mathrm{pa}(v)}). \tag{6.1}$$

When a probability distribution $\mathbb{P}$ satisfies (6.1), we say that $\mathbb{P}$ *factorizes* according to G. When $\mathbb{P}$ factorizes according to G, then $\mathbb{P}\big(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}\big) = p_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$. The following fundamental property holds: Given a collection of conditional distributions $\{p_{v|\mathrm{pa}(v)}\}_{v \in V}$, Equation (6.1) uniquely defines a probability distribution on $\mathcal{X}_V$.

Let $(V^{\mathrm{a}}, V^{\mathrm{c}}, V^{\mathrm{r}})$ be a partition of V where V^{c} is the set of *chance vertices*, V^{a} is the set of *decision vertices*, and V^{r} is the set of *utility vertices*. The utility vertices have no descendants. For ease of notation we denote by V^{s} the union of V^{c} and V^{r}. Letters a, r, and s respectively stand for action, reward, and stochastic in V^{a}, V^{r}, and V^{s}. An *influence diagram* is a directed acyclic graph $G = (V, A)$ together with a partition $V = V^{\mathrm{a}} \cup V^{\mathrm{c}} \cup V^{\mathrm{r}}$. For convenience, we will sometimes denote an influence diagram by $G = (V^{\mathrm{s}}, V^{\mathrm{a}}, A)$. Consider a set of conditional probability distributions $\mathfrak{p} = \{p_{v|\mathrm{pa}(v)}\}_{v \in V^{\mathrm{c}} \cup V^{\mathrm{r}}}$, and a collection of *reward functions* $\mathbf{r} = \{r_v\}_{v \in V^{\mathrm{r}}}$, with $r_v : \mathcal{X}_v \to \mathbb{R}$. We define a *Parametrized Influence Diagram* (PID) as the quadruplet $(G, \mathcal{X}_V, \mathfrak{p}, \mathbf{r})$. This notion has not been introduced in the literature. We introduce it to distinguish properties due to the parametrization from properties due to the graph itself. We will sometimes refer the parameters $(\mathcal{X}_V, \mathfrak{p}, \mathbf{r})$ as $\boldsymbol{\rho}$ for conciseness.

Let Δ_v denote the set of conditional probability distributions $\delta_{v|\mathrm{pa}(v)}$ on $\mathcal{X}_v$ given $\mathcal{X}_{\mathrm{pa}(v)}$. Given a collection of conditional probability distributions $\mathfrak{p}$, a *strategy* $\boldsymbol{\delta}$ in $\Delta = \prod_{v \in V^{\mathrm{a}}} \Delta_v$ uniquely defines a probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ on $\mathcal{X}_V$ through

$$\mathbb{P}_{\boldsymbol{\delta}}(X_V = x_V) = \prod_{v \in V^{\mathrm{s}}} p_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) \prod_{v \in V^{\mathrm{a}}} \delta_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}). \tag{6.2}$$

The vector $\delta_{v|\mathrm{pa}(v)}$ of vertex v in V^{a} is the *policy* in v. Note that the probability $\mathbb{P}_{\boldsymbol{\delta}}$ factorizes according to G since it satisfies (6.1). Let $\mathbb{E}_{\boldsymbol{\delta}}$ denote the expectation of X_V according to $\mathbb{P}_{\boldsymbol{\delta}}$. We denote by MEU(G,$\boldsymbol{\rho}$) the *Maximum Expected Utility* problem associated to the PID $(G, \boldsymbol{\rho})$, which is defined as follows:

$$\max_{\boldsymbol{\delta} \in \Delta} \quad \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{v \in V^{\mathrm{r}}} r_v(X_v)\Big]. \tag*{MEU$(G,\boldsymbol{\rho})$}$$

A strategy $\boldsymbol{\delta} \in \Delta^{\mathrm{d}} \subset \Delta$ is *deterministic* if, for every $v \in V^{\mathrm{a}}$, and every $x_v, x_{\mathrm{pa}(v)} \in \mathcal{X}_v \times \mathcal{X}_{\mathrm{pa}(v)}$, $\delta_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$ is a Dirac measure. Hence, every strategies $\boldsymbol{\delta}$ in Δ^{d} satisfies

$$\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) \in \{0,1\}, \quad \forall x_{\mathrm{fa}(v)} \in \mathcal{X}_{\mathrm{fa}(v)}, \forall v \in V^{\mathrm{a}}. \tag{6.3}$$

It is well known that there always exists an optimal solution to MEU$(G, \boldsymbol{\rho})$ that is deterministic [88, Lemma C.1].

Solving MEU$(G, \boldsymbol{\rho})$ is difficult for two reasons. First, evaluating a given strategy is already difficult. The difficulty of evaluating a strategy is the difficulty of solving the *inference problem* on the underlying probabilistic graphical model. Given a feasible strategy $\boldsymbol{\delta}$, a subset of vertices $C \subseteq V$ and x_C in $\mathcal{X}_C$, the inference problem consists in computing the marginal probability of x_C according to $\mathbb{P}_{\boldsymbol{\delta}}$, i.e., $\mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C)$. This a special case of the inference problem in a directed graphical model and it is well-known that such a problem is NP-hard [30]. A good indicator of the difficulty of solving the inference problem is the *treewidth* of the graph [129]. Mauá et al. [98] showed that when the underlying graph has a bounded treewidth, the inference problem becomes polynomial, and then evaluating a strategy becomes also polynomial.

Second, optimizing over the set of strategies Δ is also difficult. Mauá et al. [98, Theorem 4] showed that even with a treewidth of 2, MEU$(G, \boldsymbol{\rho})$ is NP-hard. It means that even when the inference problem is polynomial, MEU$(G, \boldsymbol{\rho})$ is NP-hard.

Remark 8. A common practice in the literature is to define the reward function of the utility vertices $v \in V^{\mathrm{r}}$ as deterministic functions $f(x_{\mathrm{pa}(v)})$ of their parents. This case can of course be modeled in our setting: for each v in V^{r}, it suffices to define state spaces $\mathcal{X}_v = \mathcal{X}_{\mathrm{pa}(v)}$, conditional probabilities $p_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$ to be equal to 1 if $x_v = x_{\mathrm{pa}(v)}$ and 0 otherwise, and reward functions $r(x_v) := f(x_v)$. △

Remark 9. In an influence diagram $G = (V^{\mathrm{s}} \cup V^{\mathrm{a}}, A)$, one says that there is *perfect recall* of previous actions, when, there exists an ordering of V^{a}, say $\{v_1, \ldots, v_m\}$, consistent with the partial order defined by the directed acyclic graph G, such that the set of parents of each decision vertex contains the preceding decision vertices and their parents in this ordering, that is, $\mathrm{fa}(v_j) \subseteq \mathrm{pa}(v_i)$ for any $j < i$. In the absence of perfect recall, many authors have used the expression *Limited Memory Influence Diagram* (LIMID) to characterize the corresponding influence diagram [81]. In this thesis, we consider the general case of LIMIDs but we refer to them as *Influence Diagrams* throughout the thesis, following the convention adopted in Koller and Friedman [75, Chapter 23]. Many natural situations such as POMDPs have the perfect-recall property. △

6.1.2 Examples

Now we introduce some examples, shown in Figure 6.1, and Figure 6.2, which illustrate how we can model stochastic optimization problems using influence diagrams. In particular, the POMDPs with memoryless policy presented in Chapter 4 is a special case of influence diagrams and P_{ml} can be read as MEU$(G, \boldsymbol{\rho})$ on POMDPs.[1] To represent an influence diagram we use the convention of Koller and Friedman [75]: the chance vertices, decision vertices and utility

[1]While $\boldsymbol{\delta}$ is called (memoryless) policy in the literature on POMDPs, $\boldsymbol{\delta}$ is called strategy in the literature on influence diagrams (see e.g. [75, Chap. 23])

vertices are respectively represented by circles, squares and diamond.

Example 4. Consider a maintenance optimization problem in which at time t a machine is in degradation state S_t. The action A_t taken by the decision maker according to the current state is typically a binary decision that is to either perform maintenance on it or not. The problem is considered over a finite horizon with scheduled maintenance slots. State and decision together lead to a new (random) state S_{t+1}, and the triple (S_t, A_t, S_{t+1}) induces a reward R_t. This is an example of an MDP which is probably the simplest type of influence diagram, represented in Figure 6.1a. In Part I, we described a more complex problem, where the actual state S_t of the machine is often not known, and the decision maker has access instead to an observation O_t that only carries partial information about the state, which leads to a POMDP. As illustrated in Figure 6.1b, we model the decision using a memoryless policy, i.e., the decision A_t is taken based on observation O_t. $\triangle$

Example 5. Figure 6.2a depicts an influence diagram modeling the media investment strategy of a political party for the next elections. The national committee starts in a_n by deciding how much to invest into national media coverage and which budget it gives to regional committees. Based on the national popularity rating v_n after the interventions on national media, regional committees i decide which fraction a_{ri} of their funds they allocate to regional media and local committees. Based on regional popularity rating v_{ri} after the interventions on regional media, each local committee j decides how much to invest in local meetings and local media $a_{\ell j}$. The goal consists in maximizing the expected total number of local elections $r_{\ell j}$ won. $\triangle$

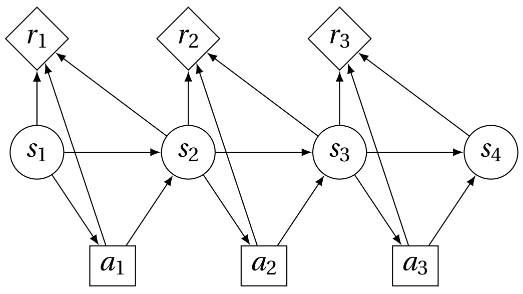
Example 6. Consider two chess players : Alice and Bob. They are used to play chess and for each game they bet a symbolic coin. However, they can decline to play. Suppose that Alice wants to play chess every day.[2] In this context, the decision maker is Bob. On day t, Alice has a current confidence level S_t. The day of the game, based on her current confidence level, Alice has a certain level of motivation denoted by M_t. When Bob meets with Alice, Bob makes the decision to play depending on Alice's demeanor, denoted U_t, which depends on her current level of motivation. Then Bob can accept or decline the challenge, and his decision is denoted by A_t. We denote by V_t the winner (getting a reward r_t). If Bob declines the challenge, there is no winner and no reward. Then, Alice's next confidence level is affected by the result of the game and her previous confidence level. This stochastic decision problem can be modeled as the maximum expected utility problem in the influence diagram shown in Figure 6.2b. $\triangle$

6.2 Junction Trees and moments

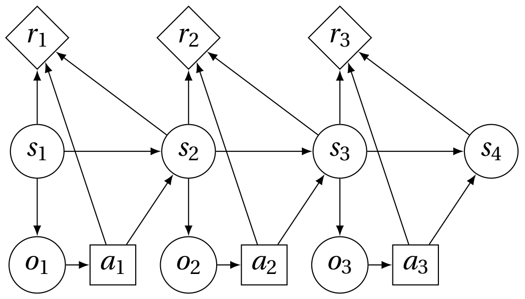
6.2.1 Junction Trees and Rooted Junction Trees (RJTs)

In order to state the main results of Part II, we recall the notion of junction tree and we introduce the new notion of Rooted Junction Tree (RJT). The *moralized graph* $G' = (V', E')$ of a directed graph $G = (V, A)$ is the undirected graph defined by $V' = V$ and $(u, v) \in E'$ if $(u, v) \in A$ or $\mathrm{ch}(u) \cap \mathrm{ch}(v) \neq \emptyset$. We denote by $M(G) = (V, M(A))$ the moralized graph of G. By definition, the families of G are cliques of $M(G)$.

[2] If Alice did not want to play every day, we would also need to model her decisions. In that case, Bob and Alice would have different objectives and we would need to use a Multi Agent Influence Diagram [76]. However, since Alice wants to play chess every day, her decisions do not need to be taken into account, and we can model the problem as an influence diagram.

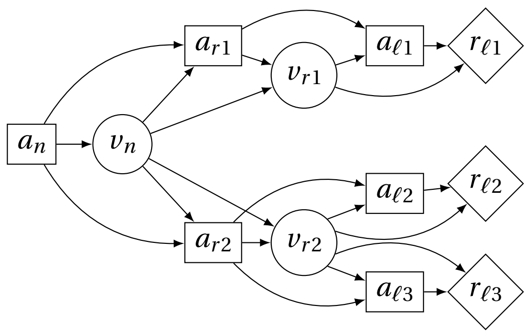


(a) A Markov decision process (MDP)

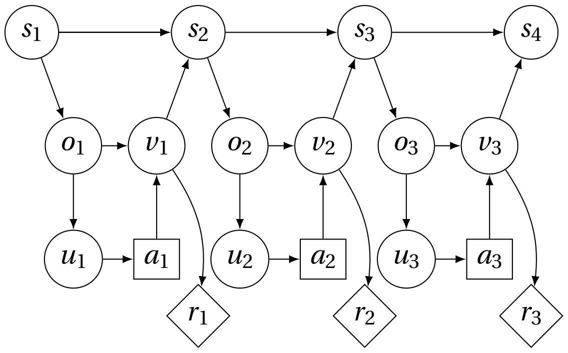


(b) A Partially Observable Markov Decision Process (POMDP) with memoryless policy

Figure 6.1 – Influence diagram examples, where we represent chance vertices (V^{s}) in circles, decision vertices (V^{a}) in rectangles, and utility vertices (V^{r}) in diamonds.



(a) Investment for local elections



(b) Bob and Alice chess game

Figure 6.2 – influence diagrams of Examples 5 and 6.

Junction tree. Let $G = (V, A)$ be a directed acyclic graph. An undirected graph $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} \subseteq 2^V$ is a *junction tree* on G if:

(i) $\bigcup_{C \in \mathcal{V}} C = V$.
(ii) For every edge in $(u, v) \in M(A)$, there is $C \in \mathcal{V}$ such that $\{u, v\} \subseteq C$.
(iii) $\mathcal{T}$ is a tree.
(iv) $\mathcal{T}$ satisfies the *running intersection property*, i.e., given two vertices C_1 and C_2 in $\mathcal{V}$, any vertex C on the unique undirected path from C_1 to C_2 in $\mathcal{T}$ satisfies $C_1 \cap C_2 \subseteq C$

Rooted Junction Tree. Let $G = (V, A)$ be a directed acyclic graph and let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be a junction tree on G together with a root. For any $v \in V$, let $\mathcal{T}_v$ be the subgraph of $\mathcal{T}$ induced by the vertices C of $\mathcal{V}$, which are also called *clusters*, containing v. The running intersection property ensures that $\mathcal{T}_v$ is a tree. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be the *directed rooted tree* obtained by orienting the edges of the tree $\mathcal{T}$ from the root to the leafs. It also defines a rooted subtree $\mathcal{G}_v = (\mathcal{V}_v, \mathcal{A}_v)$ whose underlying graph is the subtree $\mathcal{T}_v$. Let C_v denote the root of $\mathcal{G}_v$.

Definition 6.1. *A* Rooted Junction Tree *(RJT) on a directed acyclic graph $G = (V, A)$ is a directed rooted tree with vertices in 2^V, such that*

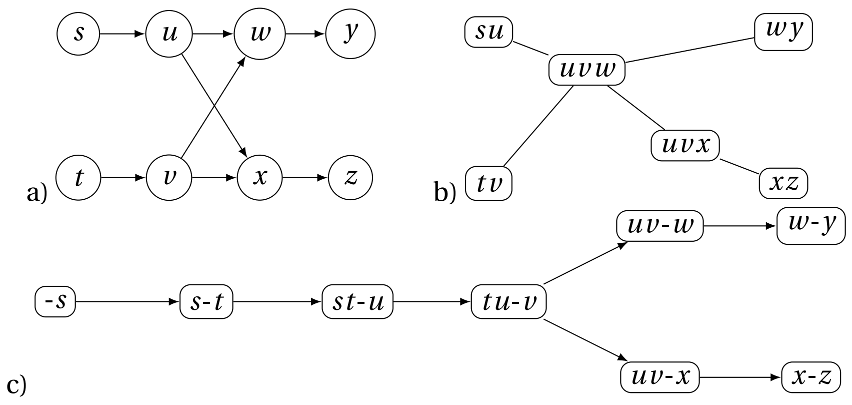


Figure 6.3 – a) A directed graph G, b) a junction tree on G, and c) a gradual rooted junction tree on G, where, for each cluster C, we indicate on the left part of the labels the vertices of $C\backslash\mathrm{offspring}(C)$, and on the right part the vertices of $\mathrm{offspring}(C)$.

(i) *its underlying undirected graph is a junction tree on G,*
(ii) *for all $v \in V$, we have* $\mathrm{fa}(v) \subseteq C_v$.

Let $\mathcal{G}$ be an RJT on G, and v a vertex of V. Given $C \in \mathcal{V}$, let the *offspring* of C be defined by $\mathrm{offspring}(C) = \{v \in V : C_v = C\}$, where C_v is the above-defined root-cluster of v, and let $\check{C}$ denote $C\backslash\mathrm{offspring}(C)$. We say that an RJT is a *gradual RJT* if for all v in V, $\mathrm{offspring}(C_v) = \{v\}$. Note that by adding vertices to an RJT, we can always turn it into a gradual RJT. Indeed, suppose that $\mathrm{offspring}(C) = \{v_1, \dots, v_k\}$, where $v_1, \dots, v_k$ are listed in a topological order. It suffices to replace the vertex C by $C_1 \to C_2 \to \cdots \to C_k$ where $C_i = C\backslash\{v_{i+1}, \dots, v_k\}$, with an arc from the parent of C to C_1 and arcs from C_k to the children of C. All the results in this thesis are more simply written with gradual RJTs even though they could have been written with RJTs. See Figure 6.3 for an example of junction tree and RJT. Given a vertex C in $\mathcal{V}$, we refer as $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{A}_C)$ the directed subtree of $\mathcal{G}$ rooted at vertex C.

In Chapter 7, we will describe the main properties of the RJT and the benefits of introducing it to model $\mathrm{MEU}(G, \boldsymbol{\rho})$.

The *width* of a junction tree $\mathcal{T} = (\mathcal{V}, \mathcal{A})$ corresponds to its maximal cluster size minus one and is denoted by $w(\mathcal{T})$, i.e., $w(\mathcal{T}) = \max_{C \in \mathcal{V}} |C| - 1$. The width of a rooted junction tree $\mathcal{G}$ is the width of its underlying undirected graph and is also denoted by $w(\mathcal{G})$. The *treewidth* (resp. *rooted treewidth*) of a graph G is the minimum width over all possible junction trees (resp. RJTs) of G. We denote respectively by $w^*(G)$ and $w^*_{\mathrm{rt}}(G)$ the treewidth of G and the rooted treewidth of G. Although $(\{V\}, \emptyset)$ is an RJT and many others gradual RJTs can be build easily, the concept of RJT has only practical interest if it is possible to construct RJTs with small width. In Section 7.5.1, we introduce an heuristic algorithm that builds an gradual RJT with a controlled width.

6.2.2 The moments on RJTs

To solve $\mathrm{MEU}(G, \boldsymbol{\rho})$ we introduce the notion of moments on a rooted junction tree of an influence diagram. This moments will be use as variables in the next sections. Let $G = (V, A)$ be an influence diagram, and $\boldsymbol{\rho}$ a parametrization on G. Let $\boldsymbol{\delta}$ be a feasible strategy on G. The *moment* μ_C of a subset of variables X_C with $C \subseteq V$ corresponds to the expected value of the indicator function of a value x_C in $\mathcal{X}_C$ according to $\mathbb{P}_{\boldsymbol{\delta}}$, i.e.,

$$\mu_C(x_C) := \mathbb{E}_{\boldsymbol{\delta}}\left[\mathbb{1}_{\{X_C = x_C\}}(x_C)\right] = \mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C).$$

We introduce the notation $\boldsymbol{\mu} = \left(\mu_C(x_C)\right)_{x_C \in \mathcal{X}_C, C \in \mathcal{V}}$. Given an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ of G and a parametrization $\boldsymbol{\rho}$ on G, we introduce the *set of achievable moments* on $\mathcal{G}$ of G by a strategy in Δ:

$$\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho}) = \left\{\boldsymbol{\mu} \in \mathbb{R}^{\prod_{C \in \mathcal{V}} \mathcal{X}_C} :\ \exists \boldsymbol{\delta} \in \Delta,\ \forall C \in \mathcal{V},\ \mu_C(x_C) = \mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C)\right\}.$$

Similarly, we introduce the *set of achievable deterministic moments* $\mathcal{M}^{\mathrm{d}}_{\mathcal{G}}(G, \boldsymbol{\rho})$ where we replaced Δ by Δ^{d} above. When $\boldsymbol{\rho}$ and $\mathcal{G}$ are clear from the context, we write more compactly $\mathcal{M}(G)$ and $\mathcal{M}^{\mathrm{d}}(G)$.

6.2.3 The value functions on RJTs

We introduce new variables to build an alternative integer program. Let $G = (V, A)$ be an influence diagram, $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ a gradual RJT of G, and $\boldsymbol{\rho}$ a parametrization on G. For convenience we extend the definition of the reward function on $\mathcal{G}$ as follows

$$r_C(x_C) = \begin{cases} r_v(x_v) \text{ if } C = C_v \text{ and } v \in V^{\mathrm{r}}, \\ 0 \text{ otherwise.} \end{cases} \tag{6.4}$$

Let $\boldsymbol{\delta}$ be a feasible strategy on G. The *value function* λ_C of a subset of variables X_C with $C \subseteq V$ corresponds to the expected reward on the subtree $\mathcal{G}_C$ induced by root C given a value x_C in $\mathcal{X}_C$ according to $\mathbb{P}_{\boldsymbol{\delta}}$, i.e.,

$$\lambda_C(x_C) := \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C' \in \mathcal{V}_C} r_{C'}(x_{C'}) | X_C = x_C\Big].$$

We introduce the notation $\boldsymbol{\lambda} = (\lambda_C(x_C))_{x_C \in \mathcal{X}_C, C \in \mathcal{V}}$. Given an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ of G and a parametrization $\boldsymbol{\rho}$ on G, we introduce the *set of achievable value functions* on $\mathcal{G}$ by a strategy in Δ as $\mathcal{F}_{\mathcal{G}}(G, \boldsymbol{\rho})$ as follows:

$$\mathcal{F}_{\mathcal{G}}(G, \boldsymbol{\rho}) = \left\{\boldsymbol{\lambda} \in \mathbb{R}^{\prod_{C \in \mathcal{V}} \mathcal{X}_C} :\ \exists \boldsymbol{\delta} \in \Delta,\ \forall C \in \mathcal{V}, \lambda_C(x_C) = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C' \in \mathcal{V}_C} r_{C'}(x_{C'}) | X_C = x_C\right]\right\}.$$

Similarly, we introduce the *set of achievable deterministic value functions* $\mathcal{F}^{\mathrm{d}}_{\mathcal{G}}(G, \boldsymbol{\rho})$ where we replaced Δ by Δ^{d} above. When $\boldsymbol{\rho}$ and $\mathcal{G}$ are clear from the context, we write more compactly $\mathcal{F}(G)$ and $\mathcal{F}^{\mathrm{d}}(G)$.

6.3 Main results

Now we state the main results of Part II. In this section, we consider a PID $(G, \boldsymbol{\rho})$. We denote by $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ an gradual RJT on G. To keep notations light in this chapter, we write the type of sum $\sum_{x_{C \backslash C'}} \mu_C(x_C)$ more compactly as $\sum_{x_{C \backslash C'}} \mu_C$ for any vertex C, C' in $\mathcal{V}$ and $x_{C \backslash C'} \in \mathcal{X}_{C \backslash C'}$. In addition, we introduce the notation $\langle r_v, \mu_v \rangle = \sum_{x_v \in \mathcal{X}_v} r_v(x_v)\mu_v(x_v)$ for any vertex v in V^{r}.

6.3.1 Integer programs using moments on $\mathcal{G}$

Non linear program for MEU$(G,\boldsymbol{\rho})$. We now introduce the following non linear program.

$$
\begin{aligned}
\max_{\boldsymbol{\mu},\boldsymbol{\delta}} \quad & \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle & & \text{(6.5a)}\\
\text{s.t.} \quad & \sum_{x_C} \mu_C = 1, & \forall C \in \mathcal{V} & \quad \text{(6.5b)}\\
& \sum_{x_{C\backslash C'}} \mu_C = \sum_{x_{C'\backslash C}} \mu_{C'}, & \forall \big(C,C'\big)\in\mathcal{A} & \quad \text{(6.5c)}\\
& \mu_{\check{C}_v} = \sum_{x_v} \mu_{C_v} & \forall v \in V & \quad \text{(6.5d)}\\
& \mu_{C_v} = \mu_{\check{C}_v} \mathrm{p}_{v|\mathrm{pa}(v)} & \forall v \in V^{\mathrm{s}} & \quad \text{(6.5e)}\\
& \mu_{C_v} = \mu_{\check{C}_v} \delta_{v|\mathrm{pa}(v)} & \forall v \in V^{\mathrm{a}} & \quad \text{(6.5f)}\\
& \boldsymbol{\mu} \geqslant 0, \boldsymbol{\delta} \in \Delta & & \text{(6.5g)}
\end{aligned}
$$

In Program (6.5), all the equalities should be understood functionally, e.g., $\mu_{C_v} = \mu_{\check{C}_v}\mathrm{p}_{v|\mathrm{pa}(v)}$ for all $v \in V^{\mathrm{s}}$ means that $\mu_{C_v}(x_{C_v}) = \mu_{\check{C}_v}(x_{\check{C}_v})\, p_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$, $\forall x_{C_v} \in \mathcal{X}_{C_v}$, for all $v \in V^{\mathrm{s}}$. We will use such functional (in)equalities throughout this dissertation.

Now we state the first result of Part II which will be proved in Chapter 8. Thanks to the properties of probability distributions, given a strategy $\boldsymbol{\delta}$, the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$ satisfies the constraints of NLP (6.5). Conversely, given a feasible solution of NLP (4.1), Theorem 6.1 ensures that $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$.

Theorem 6.1. *Let $(\boldsymbol{\mu},\boldsymbol{\delta})$ be a feasible solution of NLP* (6.5)*. Then $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$. Furthermore, $\big(\boldsymbol{\mu},\boldsymbol{\delta}\big)$ is an optimal solution of NLP* (6.5) *if and only if $\boldsymbol{\delta}$ is an optimal strategy of* MEU$(G,\boldsymbol{\rho})$*. In particular, NLP* (6.5) *and* MEU$(G,\boldsymbol{\rho})$ *have the same optimal value.*

As an immediate consequence, Theorem 6.1 ensures that: a vector $\boldsymbol{\mu}$ belongs to the set of achievable moments $\mathcal{M}(G)$ if and only if there exists a strategy $\boldsymbol{\delta}$ in Δ such that $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a feasible solution of NLP (6.5). As we will see in Chapter 7, the key assumption ensuring the validity of Theorem 6.1 is the fact that $\mathcal{G}$ is an RJT. Note that Theorem 6.1 extends Theorem 4.1 (in Chapter 4), to any influence diagrams. If we write the NLP (6.5) on POMDP with memoryless policy of Chapter 4 we obtain NLP (4.1).

Mixed-integer linear formulation for MEU$(G,\boldsymbol{\rho})$. The nonlinearity of NLP (6.5) comes from bilinear constraints (6.5f). Given two continuous variables $z, y \geqslant 0$ and $x \in \{0,1\}$, we introduce the notation McCormick$\big(z = xy\big)$ which indicates that we replace the bilinear constraint $z = xy$ by the McCormick's linear inequalities [100]. Since there always exists an optimal strategy of MEU$(G,\boldsymbol{\rho})$ that is deterministic, we can turn NLP (6.5) into the following MILP using

McCormick's inequalities without changing its optimal value.

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\delta}} \quad & \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle \\ \text{s.t.} \quad & \boldsymbol{\mu} \text{ satisfies (6.5b) – (6.5e)} \\ & \mathrm{McCormick}\big(\mu_{C_v} = \mu_{\check{C}_v}\delta_{v|\mathrm{pa}(v)}\big) \qquad \forall v \in V^{\mathrm{a}} \\ & \boldsymbol{\delta} \in \Delta^{\mathrm{d}} \end{aligned} \tag{6.6}$$

It follows from Theorem 6.1 that $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a feasible solution of MILP (6.6) if and only if $\boldsymbol{\mu}$ belongs to $\mathcal{M}^{\mathrm{d}}(G)$. Note that MILP (6.6) generalizes MILP (4.7) described in Section 4.1.2 to any influence diagrams. To write the McCormick's inequalities, we have to compute lower and upper bounds on the moments $\boldsymbol{\mu}$, that only depend on $\boldsymbol{\rho}$. The natural lower and upper bounds on the moments induced by any strategy $\boldsymbol{\delta}$ are respectively 0 and 1. In Chapter 8, we describe an approach to derive tighter bounds on the moments of a distribution that strengthen the linear relaxation of MILP 6.6.

The approach obviously has limitations. Indeed, any exact method to solve $\mathrm{MEU}(G,\boldsymbol{\rho})$ must compute the exact value of $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\big]$ when it evaluates a strategy $\boldsymbol{\delta}$. Since the exact algorithms to solve the inference problem are exponential in the treewidth $w^*(G)$, this type of method is limited in practice to graphs with moderate treewidth. The approach to solving $\mathrm{MEU}(G,\boldsymbol{\rho})$ relies on the rooted junction trees that we introduced, and is therefore practically limited to influence diagrams with moderate rooted treewidth $w^*_{\mathrm{rt}}(G)$. This is an additional limitation since $w^*_{\mathrm{rt}}(G)$ can be significantly larger than $w^*(G)$. Indeed, even if $w^*(G)$ is bounded, $w^*_{\mathrm{rt}}(G)$ can be unbounded. We however show that the approach works well on applications for which $w^*_{\mathrm{rt}}(G)$ is of the same order of magnitude as $w^*(G)$ like those considered in Examples 4, 5 and 6.

6.3.2 Valid cuts for the MILP

Now we introduce valid inequalities for MILP (6.6). The technique we introduce here systematizes our approach in Section 4.2 to construct valid inequalities for MILP (4.7). Actually using a suitable gradual RJT for POMDP with memoryless policies we could recover exactly the valid inequalities (4.8).

Let $G = (V, A)$ be an influence diagram and C a subset of vertices in V. A set of variables X_D such that $D \subseteq C$ is *strategy independent* set in C if it satisfies the following property:

For every parametrization $\boldsymbol{\rho}$ such that $(G,\boldsymbol{\rho})$ is a PID, $\mathbb{P}_{\boldsymbol{\delta}}(X_D|X_{C\backslash D})$ does not depend on $\boldsymbol{\delta}$.

For convenience, we say that D is strategy independent in C when X_D is strategy independent in C. It turns out the following lemma establishes a fundamental stability property.

Lemma 6.2. *Let $D, D' \subseteq C$. If D and D' are strategy independent in C, then $D \cup D'$ is strategy independent in C.*

Lemma 6.2 ensures the existence and uniqueness of the largest inclusion-wise strategy independent set. We denote by $C^{\perp\!\!\!\perp}$ such a subset of C. We denote by $C^{\not\perp\!\!\!\perp}$ the set $C\backslash C^{\perp\!\!\!\perp}$. The set

$C^{\perp\!\!\!\perp}$ can be empty. In Section 8.2.2, we give a full characterization of the set $C^{\perp\!\!\!\perp}$ from which we obtain the following Proposition:

Proposition 6.3. *Given a set C in V, $C^{\perp\!\!\!\perp}$ can be computed in $O\big(|C|(|V|+|A|)\big)$.*

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a gradual RJT of G. We introduce the following linear equalities:

$$\mu_C = \mu_{C^{\not\perp\!\!\!\perp}} p_{C^{\perp\!\!\!\perp}|C^{\not\perp\!\!\!\perp}}, \quad \forall C \in \mathcal{V}, \tag{6.7}$$

Now, we state the second result of Part II.

Proposition 6.4. *Equalities* (6.7) *are valid for MILP* (6.6).

In addition, we will show in Chapter 8 that the valid cuts (6.7) are the strongest linear equalities we can obtain of the form $\mu_C = \mu_{C\backslash D} p_{D|C\backslash D}$ where $D \subseteq C$. Again, note that Proposition 6.4 extends Proposition 4.3 to any influence diagrams. Given a carefully chosen RJT on POMDP with memoryless policies, equalities (6.7) correspond to equalities (4.8).

6.3.3 Integer programs using value functions

Without loss of generality, we assume that $\mathcal{G}$ has a single root vertex. Otherwise, we add an isolated dummy vertex v_0 to the original influence diagram, which we allow us to extend the RJT by adding the cluster vertex $C_0 = \{v_0\}$. If $v_0 \notin V^{\mathrm{s}}$, then we can add a vertex v_0' and a cluster vertex $C_0' = \{v_0'\}$, the random variable $X_{v_0'}$ equals to 1 almost surely, and an arc (C_0', C_0). Hence, without loss of generality, we assume that $v_0 \in V^{\mathrm{s}}$. For simplicity, we denote by $x_0 := x_{v_0}$.

Non linear program for $\mathrm{MEU}(G, \boldsymbol{\rho})$**.** We now introduce the following non linear program.

$$\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \quad \langle p_0, \lambda_{C_0} \rangle \tag{6.8a}$$

$$\text{s.t.} \quad \lambda_{C_v} = r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in V^{\mathrm{s}} \cap \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in V^{\mathrm{s}} \cap \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} \delta_{u|\mathrm{pa}(u)}, \quad \forall v \in V \tag{6.8b}$$

$$\boldsymbol{\delta} \in \Delta \tag{6.8c}$$

Now we state the third result of Part II, which will be proved in Chapter 8. Thanks to the properties of the conditional expectation, the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$ satisfies the constraints of NLP (6.8). Conversely, given a feasible solution $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ of NLP (6.8), Theorem 6.5 ensures that $\boldsymbol{\lambda}$ is the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$.

Theorem 6.5. *Let $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ be a feasible solution of NLP* (6.8)*. Then $\boldsymbol{\lambda}$ is the vector of value functions of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$. Furthermore, $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ is an optimal solution of NLP* (6.8) *if and only if $\boldsymbol{\delta}$ is an optimal policy of* $\mathrm{MEU}(G, \boldsymbol{\rho})$*. In particular, NLP* (6.8) *and* $\mathrm{MEU}(G, \boldsymbol{\rho})$ *have the same optimal value.*

As an immediate consequence, Theorem 6.5 ensures that: a vector $\boldsymbol{\lambda}$ belongs to the set of achievable value functions $\mathcal{F}(G)$ if and only if there exists a strategy $\boldsymbol{\delta}$ in Δ such that $(\boldsymbol{\lambda}, \boldsymbol{\delta})$

is a feasible solution of NLP (6.8). Theorem 6.5 extends Theorem 8.17 (in Chapter 4) to any influence diagrams. If we write the NLP (6.8) on POMDP with memoryless policy of Chapter 4, we obtain NLP (4.18).

Mixed-integer linear formulation for $\mathrm{MEU}(G,\boldsymbol{\rho})$**.** The nonlinearity of NLP (6.8) comes from the bilinear terms $\lambda_{C_u}\delta_{u|\mathrm{pa}(u)}$ in constraints (6.8b). Since there always exists an optimal strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$ that is deterministic, we can turn NLP (6.8) into a MILP using McCormick's inequalities. To do so we introduce variables $\boldsymbol{\alpha} = (\alpha_C(x_C))_{x_C\in\mathcal{X}_C, C\in\mathcal{V}}$ and the following MILP.

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\delta}} \quad & \langle p_0, \lambda_{C_0}\rangle \\
\text{s.t.} \quad & \lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in V^{\mathrm{s}}\cap\mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in V^{\mathrm{s}}\cap\mathrm{ch}(C_v)}} \sum_{x_u} \alpha_{C_u}, && \forall v\in V \\
& \mathrm{McCormick}\big(\alpha_{C_v} = \lambda_{C_v}\delta_{v|\mathrm{pa}(v)}\big), && \forall v \in V^{\mathrm{a}} \\
& \boldsymbol{\delta}\in\Delta^{\mathrm{d}}
\end{aligned} \tag{6.9}$$

Given a strategy $\boldsymbol{\delta}\in\Delta$, it follows from Theorem 6.5 that $(\boldsymbol{\lambda},\boldsymbol{\delta})$ is a feasible solution of MILP (6.6) if and only if $\boldsymbol{\lambda}$ belongs to $\mathcal{F}^{\mathrm{d}}(G)$. Note that MILP (6.9) generalizes MILP (4.21) described in Section 4.1.2 to any influence diagrams. To write McCormick's inequalities, it requires to compute lower and upper bounds on the value functions that only depend on parameters $\boldsymbol{\rho}$. The natural lower and upper bounds on the value function induced by any strategy $\boldsymbol{\delta}$ are respectively $\sum_{v\in V^{\mathrm{r}}}\min_{x_v} r_v(x_v)$ and $\sum_{v\in V^{\mathrm{r}}}\max_{x_v} r_v(x_v)$. In Chapter 8 we show how to improve these bounds.

6.3.4 Polynomial cases of Influence Diagrams

Soluble influence diagrams. The third result in Part II is based on the notion of *soluble* influence diagrams. To define a soluble influence diagram, we need to introduce the notion of local optimum strategy. Consider a PID $(G,\boldsymbol{\rho})$ with $G=(V,A)$, $V=V^{\mathrm{s}}\cup V^{\mathrm{a}}$ and a parametrization $\boldsymbol{\rho}$. Given a strategy $(\delta_u)_{u\in V^{\mathrm{a}}}$ and a decision vertex v, we denote by δ_{-v} the partial strategy $(\delta_u)_{u\in V^{\mathrm{a}}\backslash v}$. A strategy $\boldsymbol{\delta}$ is a *local optimum* if

$$\delta_v \in \operatorname{arg\,max}_{\delta'_v\in\Delta_v} \mathbb{E}_{\delta'_v,\delta_{-v}}\left(\sum_{u\in V^{\mathrm{r}}} r_u(X_u)\right) \quad \text{for each vertex } v \text{ in } V^{\mathrm{a}}.$$

It is a *global optimum* if it is an optimal solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$. The problem of finding a local optimum is "easy" in the following sense: Suppose that we have an oracle that gives us the result of the inference problem in polynomial time, then a local optimum can be computed in polynomial time [75, Proposition 23.2].

Definition 6.2. *An influence diagram G is* soluble *if for every parametrization* $\boldsymbol{\rho}$ *of G, every local optimum is a global optimum.*

Because of the remark above, $\mathrm{MEU}(G,\boldsymbol{\rho})$ is “easy” to solve when G is soluble. Alternate definitions of soluble influence diagrams are available in the literature (see Section 6.4). In the literature, soluble influence diagrams are also defined using tools of graphical models, which are introduced in Chapter 7. Thanks to these definitions, the following proposition is common knowledge in the literature.

Proposition 6.6. *Deciding if an influence diagram G is soluble can be done in polynomial time.*

Now, we state our fourth result, which provides a link between a soluble influence diagrams and MILP (6.6).

Theorem 6.7. *If G is soluble, then there exists an RJT, such that, for every parametrization $\boldsymbol{\rho}$, an optimal solution of the linear relaxation of MILP* (6.6) *with the valid inequalities* (6.7) *induces an optimal solution of* $\mathrm{MEU}(G,\boldsymbol{\rho})$ *and both problems have the same optimal values. Such an RJT can be computed in polynomial time.*

Theorem 6.7 confirms that, when the inference problem is tractable (small rooted-treewidth), solving $\mathrm{MEU}(G,\boldsymbol{\rho})$ is tractable for every parametrization $\boldsymbol{\rho}$ when G is soluble.

As mentioned for Theorem 6.1, given a parametrization $\boldsymbol{\rho}$ and a RJT $\mathcal{G}$ the set of achievable moments $\mathcal{M}_{\mathcal{G}}(G,\boldsymbol{\rho})$ fully characterizes the solutions of NLP (6.5), which is a Quadratically Constrained Quadratic Program (QCQP). Hence, the convexity of the set $\mathcal{M}_{\mathcal{G}}\big(G,\boldsymbol{\rho}\big)$ is a measure of the ability to solve (6.5). Our fifth result provides a characterization of soluble influence diagrams in terms of convexity of the set $\mathcal{M}_{\mathcal{G}}\big(G,\boldsymbol{\rho}\big)$.

Theorem 6.8. *An influence diagram G is soluble if and only if there exists an RJT $\mathcal{G}$ such that for every parametrization $\boldsymbol{\rho}$ on G, the set of achievable moments $\mathcal{M}_{\mathcal{G}}\big(G,\boldsymbol{\rho}\big)$ is a polytope. In this case, such an rooted junction tree $\mathcal{G}$ can be computed in polynomial time.*

While Theorem 6.8 provides a necessary and sufficient condition on influence diagrams for being soluble, Theorem 6.7 gives only a necessary condition. In fact, it turns out the linear relaxation of MILP (6.6) with valid inequalities (6.7) provides optimal solutions for a slightly larger class of influence diagrams, including some PID with a non-convex set of achievable moments. In Chapter 9, we describe an example showing this phenomenon.

6.3.5 Dual formulations for the linear relaxations.

The linear relaxations of MILP (6.6) with and without valid inequalities (6.7) play a key role to derive bounds. It turns out the dual of these relaxations can be formulated using value functions variables in a linear formulation which is closely related to a linear formulation of dynamic programming algorithm Using the variables representing the value functions we introduce the two following linear programs:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \langle \lambda_{C_0}, p_0 \rangle \\
\text{s.t.} \quad & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \lambda_{C_u} \qquad \forall v \in V.
\end{aligned} \tag{6.10}$$

$$\begin{aligned}\min_{\boldsymbol{\lambda}} \quad & \langle \lambda_{C_0}, p_0 \rangle \\ \text{s.t.} \quad & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\perp}}} \lambda_{C_u} p_{C_u^{\perp\perp}|C_u^{\not\perp\perp}} \qquad \forall v \in V.\end{aligned} \tag{6.11}$$

Now, we state the sixth main result of Part II.

Theorem 6.9. *The following properties hold:*

(i) *Linear program* (6.10) *is the dual of the linear relaxation of MILP* (6.6) *where variable* $\boldsymbol{\delta}$ *has been removed.*
(ii) *Linear program* (6.11) *is the dual of the linear relaxation of MILP* (6.6) *with valid inequalities* (6.7) *where variable* $\boldsymbol{\delta}$ *has been removed.*

Furthermore, the strong duality holds in both cases.

Theorem 6.9 says that there is a duality relation between the moments variables and value functions variables. In addition, it follows from Theorem 6.9 that if G is soluble, then by Theorem 6.7 the linear program (6.11) induces also an optimal solution of $\mathrm{MEU}(G, \boldsymbol{\rho})$. Linear program (6.11) is equivalent to the linear formulation of dynamic programming approach on the RJT $\mathcal{G}$. In the case of MDPs, by writing linear program (6.11) we recover exactly the well-known linear formulation of dynamic programming on MDP (see, e.g., Puterman [124]).

6.4 Bibliographical remarks

Influence diagrams were introduced by Howard and Matheson [56] [see also 57] to model stochastic optimization problems using a probabilistic graphical model framework. Originally, the decision makers were assumed to have *perfect recall* [62, 137, 141] of the past actions.

Lauritzen and Nilsson [81] relaxed this assumption and provided a simple (coordinate descent) algorithm to find a good strategy: the Single Policy Update (SPU) algorithm. By relaxing the perfect recall assumption, these authors referred the resulting influence diagrams as *limited memory influence diagrams*. However, we follow the convention of Koller and Friedman [75] who still call them influence diagrams. In general, SPU finds a locally optimal strategy in a finite number of iterations, and requires to perform exact inference, so that it is therefore limited by the treewidth [25]. Lauritzen and Nilsson [81] also introduced a notion of soluble influence diagram using tools of directed graphical models. In particular, the authors proved that being soluble is a sufficient condition for SPU to converge to an optimal solution in a finite and polynomial number of iterations. Koller and Milch [76] generalized their notion of soluble influence diagram to make this condition necessary and sufficient.

More recently, Mauá and Campos [95] and Mauá and Cozman [96] have introduced a new algorithm, *Multiple Policy Update*, which has both an exact and a heuristic version and relies on a concept of dominance to discard partial solutions. It can be interpreted as a generalization of SPU where several decisions are considered simultaneously. Later on, Khaled et al. [68] proposed a similar approach, in the spirit of Branch-and-Bound, while Liu [88] introduced heuristics based on approximate variational inference. Usually, inference computations in influence diagrams are done within ”valuation algebra” on the pair probability-utility [35, 62], which is an abstract framework that facilitates computations in graphical models [139]. Lee et al. [83] propose an inference algorithm providing upper bounds on the MEU which uses the same "valuation algebra" for influence diagrams [35]. Even if the inference computations in influence diagrams are commonly done using valuation algebra, we use instead the marginal polytope because it is useful for mathematical programming approaches [148, 156].

Finally, the problem of solving an influence diagram can be polynomially transformed into a "maximum a posteriori" (MAP) problem, which is well-known in the graphical model community. Hence, it can be solved using popular MAP solvers such as `toulbar2` [58]. For further details about the transformation, see Antonucci and Zaffalon [4], Cano et al. [21] and Maua [94].

Finding an optimal strategy for an influence diagram has been shown to be NP-hard even when restricted to influence diagrams of treewidth no greater than two, or to trees with binary variables [97, 99]. Note that even obtaining an approximate solution is also NP-hard [97].

Beyond the classical linear programming formulation for MDPs, mathematical programming formulations have been proposed for some special cases of influence diagrams, including decomposable or weakly coupled MDPs [2, 15, 17, 34, 55] and POMDP with perfect recall and short horizon [5]. As noted in the first part, the special case of POMDP with memoryless policies extends the work of Bertsimas and Mišić [15] to POMDPs since it also relies on variables that corresponds to moments or distributions. The variables of the other formulations correspond to time averages [17] or value functions [2, 34, 55], which makes these formulations harder to generalize to influence diagrams.

Credal networks are generalizations of probabilistic graphical models where the parameters of the model are not known exactly. MILP formulations for credal networks that could be applied to influence diagrams have been introduced by de Campos and Cozman [32], de Campos and Ji [33]. However, the number of variables they require is exponential in the *pathwidth*, which is non-smaller and can be arbitrarily larger than the width of the tree we are using [133, Theorem 4], and the linear relaxation of their MILP is not as good as the one of the MILP we propose, and does not yield an integer solution on soluble influence diagrams.

Examples 4, 5 and 6 are sequential decisions problems in stochastic optimization. Many different solution approaches have been proposed for these kinds of problems under different names in different academic communities. While describing these approaches is beyond the scope of this thesis, we refer the interested reader to the tutorial of Powell [122]. In particular, the literature on the POMDP example 4 has been detailed in Part I.

7 Graphical models and rooted junction tree properties

This chapter is dedicated to introduce first the tools and the intermediate results to prove Theorem 6.1, and second the algorithm mentioned in Section 6.2.1 to build a gradual RJT with a controlled width. As mentioned above, to evaluate a strategy $\boldsymbol{\delta}$ in an influence diagram, we need to encode the probability $\mathbb{P}_{\boldsymbol{\delta}}$ using a vector of moments. It raises the problem of characterizing the probability distributions factorizing on a directed graph. Once such a problem has been addressed the proof of Theorem 6.1 will be eased. Given a probability factorizing on a directed graphical model, we can derive the corresponding vector of moments and such a vector necessary satisfies some properties that we will describe in this chapter. However, the main difficulty in proving Theorem 6.1 is to understand how a vector of moments satisfying such properties is sufficient to encode a probability distribution factorizing on a directed graphical model.

This chapter is organized as follows:

- Section 7.1 introduces basic notation from graph theory.
- Section 7.2 recalls the main theorem characterizing a probability distribution factorizing on a directed graphical model, in terms of conditional independences between the random variables.
- Section 7.3 recalls the notion of junction tree and a "local" version of the conditional independences that is necessarily satisfied by a vector of moments in the marginal polytope of a junction tree. We show that these independences are not sufficient on a junction tree to ensure the factorization of the probability distribution.
- Section 7.4 introduces several properties of RJTs. In particular, we show that the local independences on a RJT of Section 7.3 are sufficient to ensure a global factorization on a directed graphical model.
- Section 7.5 introduces an algorithm that builds a gradual RJT with a controlled width. In addition, it provides a characterization of the gradual RJT built by the algorithm. This characterization will be useful for the proofs of Theorem 6.7 and Theorem 6.8.

7.1 Graph notation

This section introduces notation for graphs, which are mostly those one commonly used in the combinatorial optimization community [135]. A (simple) directed graph G is a pair (V, A)

where V is the set of vertices and $A \subseteq V^2$ the set of arcs. We write $u \to v$ when $(u,v) \in A$. Let $[k] := \{1,\dots,k\}$. A *path* is a sequence of vertices $v_1,\dots,v_k$ such that $v_i \to v_{i+1}$, for any $i \in [k-1]$. A path between two vertices u and v is called a u-v path. We write $u \overset{G}{\dashrightarrow} v$ to denote the existence of a u-v path in G, or simply $u \dashrightarrow v$ when G is clear from context. We write $u \rightleftharpoons v$ if there is an arc $u \to v$ or $v \to u$. A *trail* is a sequence of vertices $v_1,\dots,v_k$ such that $v_i \rightleftharpoons v_{i+1}$, for all $i \in [k-1]$.

A vertex u is an *ancestor* (resp. a *descendant*) of v if there exists a u-v path (resp. a v-u path). We denote respectively by $\mathrm{anc}(v)$ and $\mathrm{des}(v)$ the set of ancestors and descendants of v. Finally, let $\overline{\mathrm{anc}}(v) = \{v\} \cup \mathrm{anc}(v)$, and $\overline{\mathrm{des}}(v) = \{v\} \cup \mathrm{des}(v)$. For a set of vertices C, the parent set of C, again denoted by $\mathrm{pa}(C)$, is the set of vertices u that are parents of a vertex $v \in C$. We define similarly $\mathrm{fa}(C)$, $\mathrm{ch}(C)$, $\mathrm{anc}(C)$, $\overline{\mathrm{anc}}(C)$, $\mathrm{des}(C)$, and $\overline{\mathrm{des}}(C)$. Note that we sometimes indicate in subscript the graph according to which the parents, children, etc., are taken. For instance, $\mathrm{pa}_G(v)$ denotes the parents of v in G.

A *cycle* is a path $v_1,\dots,v_k$ such that $v_1 = v_k$. The underlying undirected graph is *connected* if there exists a path between any pair of vertices. An *acyclic* graph is a graph which has no cycle. An undirected graph is a *tree* if it is connected and acyclic. A directed graph is a *directed tree* if its underlying undirected graph is a tree. A *rooted tree* is a directed tree such that all vertices have a common ancestor referred to as the *root* of the tree.[1] In a rooted tree, all vertices but the root have exactly one parent.

7.2 Directed graphical model

In this thesis, given three random variables X, Y, Z, the notation $(X \perp\!\!\!\perp Y|Z)_{\mathbb{P}}$ stands for "X is independent from Y given Z" according to the probability distribution $\mathbb{P}$ of the random vector (X,Y,Z). We underline that $(\cdot)_{\mathbb{P}}$ corresponds to independence according to the probability distribution $\mathbb{P}$, and should not be confused with the notation $(\cdot)_G$ that is more frequently used in the literature and stands for d-separation according to the graph G.

As mentioned in Chapter 6, evaluating a strategy in an influence diagram requires to solve the inference problem in a directed graphical model, which is equivalent to compute the probability distribution $\mathbb{P}_\delta$ given a strategy $\boldsymbol{\delta}$. Since an influence diagram is a directed graphical model, we consider more generally probability distributions that factorize according to a directed graphical model. A well-known sufficient condition for a distribution to factorize as a directed graphical model is that each vertex is independent from its non-descendants given its parents as stated in the following fundamental proposition.

Theorem 7.1. *[75, Theorem 3.1, p. 62]. Let X_V be a random variable on $\mathcal{X}_V$. Then its distribution $\mathbb{P}$ factorizes according to a directed acyclic graph $G = (V,A)$, i.e., $\mathbb{P}$ satisfies* (6.1) *if and only if*

$$\big(X_v \perp\!\!\!\perp X_{V\backslash\overline{\mathrm{des}}_G(v)} | X_{\mathrm{pa}(v)}\big)_{\mathbb{P}} \quad \textit{for all } v \textit{ in } V. \tag{7.1}$$

Note that this result is sometimes considered as the counterpart of the theorem of Hammersley

[1] The probabilistic graphical model community sometimes calls a *directed tree* what we call here a *rooted tree*, and a *polytree* what we call here a *directed tree*.

and Clifford [75, Theorem 4.2, p. 116] for directed graphical models. Theorem 7.1 plays a key role in the subsequent results of this chapter.

7.3 Moments on junction trees

The goal of this section is to recall a result of probabilistic graphical model theory that explains the role of the vector of moments in directed graphical models. Such a result is key in proving Theorem 6.1. We start by introducing useful definitions required to present this result.

Given a probability distribution $\mathbb{P}$ on $\mathcal{X}_V$, we define the vector of moments $\boldsymbol{\mu} = (\mu_C(x_C))_{x_C \in \mathcal{X}_C, C \in \mathcal{V}}$ of $\mathbb{P}$ as follows:

$$\mu_C(x_C) = \sum_{x_{V\backslash C}} \mathbb{P}(X_V = x_V) \quad \forall x_C \in \mathcal{X}_C,\ C \in \mathcal{V}, \tag{7.2}$$

where $\mathcal{V}$ is a subset of 2^V. Given $\mathcal{V} \subseteq 2^V$, we say that a vector $\boldsymbol{\mu}$ derives from a probability distribution on $\mathcal{X}_V$ if there exists $\mathbb{P}$ defined on $\mathcal{X}_V$ such that $\boldsymbol{\mu}$ satisfies (7.2). We denote by $\mathbb{P}_{\boldsymbol{\mu}}$ such a probability distribution on $\mathcal{X}_V$. Given a junction tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, we define its associated *marginal polytope*

$$\mathcal{M}^0(\mathcal{T}) = \left\{ \boldsymbol{\mu} : \left| \begin{array}{l} \mu_C \geqslant 0 \quad \text{and} \quad \sum_{x_C} \mu_C(x_C) = 1, \quad \forall x_C \in \mathcal{X}_C, \forall C \in \mathcal{V}, \\ \text{and} \quad \sum_{x_{C_1\backslash C_2}} \mu_{C_1} = \sum_{x_{C_2\backslash C_1}} \mu_{C_2}, \quad \forall (C_1, C_2) \in \mathcal{E}, \end{array} \right. \right\}, \tag{7.3}$$

where, as before, $\sum_{x_{C_1\backslash C_2}} \mu_{C_1}$ is the vector $\left(\sum_{x_{C_1\backslash C_2} \in \mathcal{X}_{C_1\backslash C_2}} \mu_{C_1}(x_{C_1\backslash C_2}, x_{C_1\cap C_2})\right)_{x_{C_1\cap C_2} \in \mathcal{X}_{C_1\cap C_2}}$. The constraints $\sum_{x_{C_1\backslash C_2}} \mu_{C_1} = \sum_{x_{C_2\backslash C_1}} \mu_{C_2}$ in the definition $\mathcal{M}^0(\mathcal{T})$ are usually called *local consistency* constraints [156]. Proposition 7.2 below states that the marginal polytope of a junction tree characterizes the vector of moments deriving from a probability distribution on $\mathcal{X}_V$. For convenience, we introduce the set of *separators* $\mathcal{S} = \{C_1 \cap C_2 \mid (C_1, C_2) \in \mathcal{E}\}$.[2]

Proposition 7.2. *[156, Proposition 2.1] Let $G = (V, A)$ be a directed graphical model and $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ a junction tree of G. A vector of moments $\boldsymbol{\mu}$ belongs to $\mathcal{M}^0_{\mathcal{T}}$ if and only if $\boldsymbol{\mu}$ derives from a probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ on $\mathcal{X}_V$. Moreover, this probability distribution is unique and defined by*

$$\mathbb{P}_{\boldsymbol{\mu}}(X_V = x_V) = \frac{\prod_{C \in \mathcal{V}} \mu_C(x_C)}{\prod_{S \in \mathcal{S}} \mu_S(x_S)}.$$

In fact, $\mathcal{M}^0(\mathcal{T})$ is usually called the *local polytope* (see e.g. Wainwright and Jordan [156]) and the marginal polytope is the set of vector of moments deriving from a probability distribution over all the random variables. Thanks to Proposition 7.2, when $\mathcal{T}$ is a junction tree, the local polytope coincides with the marginal polytope.

[2]The separators are often included in the definition of the junction tree and their associated moments τ_S in the definition of the marginal polytope. We do not include them in this work, because we do not need them in our mathematical programming formulations. Adding them would increase the size of the mathematical program and downgrade the performance of the solver.

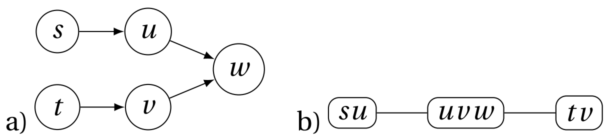


Figure 7.1 – Example where satisfying (7.4) on junction tree b) is not sufficient to ensure factorization on graph a).

7.4 Moments on rooted junction trees

7.4.1 Main properties

Thanks to Theorem 7.1, a necessary and sufficient condition for a probability distribution to factorize is to satisfy global independences (7.1). In this section, we show that we still have such a necessary and sufficient condition when we replace the global independences (7.1) by "local" independences introduced on an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ as follows: We say that a vector $\boldsymbol{\mu}$ in the marginal polytope $\mathcal{M}^0(\mathcal{G})$ satisfies "local" independences on $\mathcal{G}$ if for all $C \in \mathcal{V}$, we have

$$\left(X_v \perp\!\!\!\perp X_{C\backslash\overline{\mathrm{des}}(v)} | X_{\mathrm{pa}(v)}\right)_{\mu_C}, \text{ for all } v \text{ in } V \text{ such that } \mathrm{fa}(v) \subseteq C. \tag{7.4}$$

Theorem 7.3. *Let $\boldsymbol{\mu}$ be a vector of moments in the marginal polytope of an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ of $G = (V, A)$. If $\boldsymbol{\mu}$ satisfies* (7.4)*, then the unique probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ on X_V factorizes according to G.*

An important remark is that Theorem 7.3 is not true when we consider a junction tree instead of an RJT. Consider a probability distribution $\mathbb{P}$ factorizing according to a directed graphical model $G = (V, A)$. Given a junction tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, we obtain that: The vector of moments of $\mathbb{P}$ necessarily belongs to the marginal polytope $\mathcal{M}^0(\mathcal{T})$ and satisfies (7.4). However, the reverse is not true. Indeed, a vector $\boldsymbol{\mu}$ belonging to the marginal polytope $\mathcal{M}^0(\mathcal{T})$ and satisfying (7.4) does not ensure that the unique probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ from which $\boldsymbol{\mu}$ is derived factorizes according to G. For instance, on the junction tree of Figure 7.1.b, Equation (7.4) does not enforce the independence of u and v, which is required on the graph of Figure 7.1.a. But Theorem 7.3 ensures that property (7.4) becomes a sufficient condition under the additional assumption that $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is an RJT.

In this section we present further technical results on RJT that are useful to prove Theorem 7.3. We start with generic properties of RJT.

Proposition 7.4. *Let $\mathcal{G}$ be an RJT on G.*

1. *If there is a path from u to v in G, then there is a path from C_u to C_v in $\mathcal{G}$.*
2. *If $\overline{\mathrm{des}}_G(u) \cap \overline{\mathrm{des}}_G(v) \neq \emptyset$, then either there is a unique path from C_u to C_v or from C_v to C_u in $\mathcal{G}$.*

Proof. Let $\mathcal{G}$ be an RJT on G. Consider a vertex v of G and a vertex C of $\mathcal{G}$ containing v. Recall that $\mathcal{G}_v$ is the subtree of $\mathcal{G}$ induced by the vertices containing v. Since C is a vertex of $\mathcal{G}_v$, and by definition of C_v, there exists a C_v-C path in $\mathcal{G}$. Now consider $u \in \mathrm{pa}(v)$. Since $\mathrm{fa}(v) \subseteq C_v$, we have $u \in C_v$. Thus there exists a C_u-C_v path in $\mathcal{G}$. The first statement follows by induction along a u-v path in G.

We now show the second statement. Let w be a vertex in $\overline{\mathrm{des}}_G(u) \cap \overline{\mathrm{des}}_G(v)$, then by the first statement there exists both a C_u-C_w and a C_v-C_w path in $\mathcal{G}$. As $\mathcal{G}$ is a rooted tree, this implies the existence of either a C_u-C_v path or of a C_v-C_u path in $\mathcal{G}$. □

The following lemma is key in proving Theorem 7.3.

Lemma 7.5. *Let C, D be subsets of V such that $\mathrm{fa}(D) \subseteq C$ and $\overline{\mathrm{des}}(D) \cap C = D$. Consider a distribution μ_C on C. Suppose that for each v in D, X_v is independent from its non-descendants given its parents according to μ_C. Then, μ_C factorizes as $\mu_C = \mu_{C\backslash D} \prod_{v\in D} q_{v|\mathrm{pa}(v)}$ where $\mu_{C\backslash D} = \sum_{x_D} \mu_C$ and $q_{v|\mathrm{pa}(v)}$ is defined as $\frac{\sum_{x_{C\backslash \mathrm{fa}(v)}} \mu_C}{\sum_{x_{C\backslash \mathrm{pa}(v)}} \mu_C}$ when the denominator is non-zero, and as 0 otherwise.*

Proof. Let $\preccurlyeq$ be a topological order on C such that $u \in C\backslash D$ and $v \in D$ implies $u \preccurlyeq v$. Such a topological order exists since $\overline{\mathrm{des}}(D) \cap C = D$. We have

$$\mu_C = \mu_{C\backslash D} \prod_{v\in D} \mathbb{P}_\mu(X_v | X_u, u \in C, u \prec v) = \mu_{C\backslash D} \prod_{v\in D} \mathbb{P}_\mu(X_v | X_{\mathrm{pa}(v)}),$$

where the first equality is the chain rule and the second follows from the hypothesis of the lemma. □

Proof of Theorem 7.3. Let $\mathcal{G}$ be an RJT on G. Let $C_1, \ldots, C_n$ be a topological ordering on $\mathcal{G}$, let $C_{\leqslant i} = \bigcup_{j\leqslant i} C_j$, and $C_{<i} = C_{\leqslant_i} \backslash C_i$. Let $\boldsymbol{\tau}$ be a vector of moments satisfying the hypothesis of the theorem, and for each v in V, let $q_{v|\mathrm{pa}(v)}$ be equal to $\frac{\sum_{x_{C\backslash \mathrm{fa}(v)}} \tau_{C_v}}{\sum_{x_{C\backslash \mathrm{pa}(v)}} \tau_{C_v}}$ if the denominator is non-zero, and to 0 otherwise. We show by induction on i that

$$\mu_{C_{\leqslant i}} = \prod_{v \in C_{\leqslant i}} q_{v|\mathrm{pa}(v)} \quad \text{is such that} \quad \tau_{C_{i'}} = \sum_{x_{C_{\leqslant i} \backslash C_{i'}}} \mu_{C_{\leqslant i}} \quad \text{for all } i' \leqslant i.$$

Suppose the result true for all $j < i$, with the convention that $\mu_0 = 1$. We immediately deduce from the induction hypothesis that $\tau_{C_{i'}} = \sum_{x_{C_{\leqslant i} \backslash C_{i'}}} \mu_{C_{\leqslant i}}$ for all $i' < i$. There only remains to prove that $\tau_{C_i} = \sum_{x_{C_{<i}}} \mu_{C_{\leqslant i}}$. By definition of an RJT, we have $\mathrm{fa}(\mathrm{offspring}(C_i)) \subseteq C_i$. Proposition 7.4 implies that $\mathrm{des}(\mathrm{offspring}(C_i)) \cap C_i \subseteq \mathrm{offspring}(C_i)$. Indeed let u be in $\mathrm{des}(\mathrm{offspring}(C_i)) \cap C_i$. Then there is a C_i-C_u path as $u \in \mathrm{des}(C_i)$, and a C_u-C_i path as $u \in C_i$. Hence $C_u = C_i$ and $u \in \mathrm{offspring}(C_i)$. In addition, $\boldsymbol{\tau}$ satisfies (7.4). Hence, τ_{C_i} is a distribution on C_i such that each vertex in $\mathrm{offspring}(C_i)$ is independent from its non-descendants given its parents. By applying Lemma 7.5 with $D = \mathrm{offspring}(C_i)$, we have $\tau_{C_i} = \tau_{C_i \backslash \mathrm{offspring}(C_i)} \prod_{v \in \mathrm{offspring}(C_i)} q_{v|\mathrm{pa}(v)}$. Let C_j be the parent of C_i in $\mathcal{G}$, we have $\tau_{C_i \backslash \mathrm{offspring}(C_i)} = \sum_{x_{C_j \backslash C_i}} \tau_{C_j} = \sum_{x_{C_{<i} \backslash C_i}} \mu_{C_{<i}}$, the first equality coming from the fact that $(\tau_C)_{C \in \mathcal{V}}$ belongs to the marginal polytope, and the second from the

induction hypothesis. Thus,

$$\begin{aligned}\sum_{x_{C_{<i}}} \mu_{C_{\leqslant i}} = \sum_{x_{C_{<i}}} \prod_{v\in V_{\leqslant i}} q_{v|\mathrm{pa}(v)} &= \Big(\sum_{x_{C_{<i}\backslash C_i}} \mu_{C_{<i}}\Big) \prod_{v\in\mathrm{offspring}(C_i)} q_{v|\mathrm{pa}(v)} \\ &= \tau_{C_i\backslash\mathrm{offspring}(C_i)} \prod_{v\in\mathrm{offspring}(C_i)} q_{v|\mathrm{pa}(v)} \\ &= \tau_{C_i},\end{aligned}$$

which gives the induction hypothesis, and the theorem. □

When the RJT is gradual, Theorem 7.3 together with Lemma 7.5 gives the following corollary.

Corollary 7.6. *Let $\boldsymbol{\mu}$ be a vector of moments in the marginal polytope of a gradual RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ on $G = (V, A)$. The probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ on X_V factorizes according to G if and only if for all $v \in V$, all $x_{\mathrm{pa}(v)}$ in $\mathcal{X}_{\mathrm{pa}(v)}$ such that $\mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) \neq 0$, and all $x_{C_v\backslash\mathrm{pa}(v)}$, we have*

$$\mu_{C_v}(x_{C_v}) = \mu_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})\,\mu_{\check{C}_v}(x_{\check{C}_v}), \quad \textit{where} \quad \mu_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) := \frac{\mu_{\mathrm{fa}(v)}(x_{\mathrm{fa}(v)})}{\mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})}.$$

Jensen et al. [62, beginning of Section 4] introduced the concept of *strong junction tree* which is similar to our concept of RJT, but they do not have the suitable properties for our approach.[3]

7.5 Building a gradual RJT

As mentioned in Chapter 6, the concept of rooted junction tree has only practical interest if it is possible to construct RJTs with a small width. In this section, we introduce an algorithm that builds a gradual RJT with a controlled width. Such a gradual RJT is minimal in a certain sense we describe here. In particular, we give a characterization of the gradual RJT built by the algorithm.

7.5.1 An algorithm to build a gradual RJT

We start by introducing a necessary condition of the RJTs. Any RJT must satisfy, for all $u, v \in V$, the implication

$$\left.\begin{array}{r}\exists w \in V\ s.t.\ C_v \dashrightarrow C_w \text{ and } u \in \mathrm{fa}(w) \\ \text{and} \qquad C_u \dashrightarrow C_v\end{array}\right\} \Rightarrow u \in C_v, \tag{7.5}$$

where $C \dashrightarrow C'$ denotes the existence of a C-C' path in the RJT $\mathcal{G}$ considered. This notation will be used throughout this section. Indeed, since $u \in C_u$ and $\mathrm{fa}(w) \subset C_w$ by definition, and

[3] The concept of strong junction tree relies on the notion of *elimination ordering* for a given *influence diagram* with perfect recall. The main difference is that a strong junction tree is a notion on an influence diagram, where the set of decision vertices and their orders play a role, while RJTs rely on the underlying digraph. The notion of strong junction tree is obtained by replacing (ii) in the definition of an RJT by: "given an elimination ordering, if (C_u, C_v) is an arc, there exists an ordering of C_v that respects the elimination ordering such that $C_u \cap C_v$ is before $C_v\backslash C_u$ in that ordering." An RJT is a strong junction tree. Conversely, a strong junction tree is not necessary an RJT. Indeed, Jensen et al. [62, Figure 4] shows an example of strong junction where there is $v \in V$ such that $\mathrm{fa}(v) \subsetneq C_v$. As strong junction trees is a notion on influence diagram and not on graphs, Theorem 7.3 has no natural generalization for strong junction trees.

since $C_u \dashrightarrow C_v \dashrightarrow C_w$, the running intersection property implies $u \in C_v$. This motivates Algorithm 4, a simple gradual RJT construction algorithm which propagates iteratively elements present in each cluster vertex to their parent cluster vertex, and which thereby produces an RJT which is *minimal* in the sense that the implication in (7.5) is strengthened to an equivalence, as stated in Proposition 7.7. It turns out that the RJT produced by Algorithm 4 has been considered in the literature under the name *bucket tree* [66, Definition 5.2].[4]

Algorithm 4 Build a minimal gradual RJT given a topological order

```
1: Input $G = (V, A)$ and a topological order $\preccurlyeq$ on G
2: Initialize $C'_v = \emptyset$ for all $v \in V$ and $\mathcal{A}' = \emptyset$
3: for each vertex $v$ of $V$ taken in reverse topological order $\preccurlyeq$ do
4:     $C'_v \leftarrow \mathrm{fa}(v) \cup \bigcup_{w:(v,w)\in\mathcal{A}'} \check{C}_w$
5:     if $\check{C}'_v \neq \emptyset$ then
6:         $u \leftarrow \max_{\preccurlyeq}\left(\check{C}_v\right)$        ▷ $u$ is the maximal element of $\check{C}_v \subset V$ according to $\preccurlyeq$
7:         $\mathcal{A}' \leftarrow \mathcal{A}' \cup (u, v)$
8:     end if
9: end for
10: $\mathcal{A} \leftarrow \{(C'_u, C'_v) \mid (u, v) \in \mathcal{A}'\}$
11: Return $\mathcal{G} = \left((C'_v)_{v\in V}, \mathcal{A}\right)$
```

The algorithm proceeds as follows: Let $\preccurlyeq$ be an arbitrary topological order on G, and $\max_{\preccurlyeq} C$ denote the maximum of C for the topological order $\preccurlyeq$. The algorithm maintains a set C'_v for each vertex v, which coincide at the end of the algorithm with the vertices C_v in the RJT produced. We recall that $\check{C}'_v$ is the set $C'_v \backslash \{v\}$. As an illustration, for any topological order on the graph of the chess example of Figure 6.2b, Algorithm 4 produces the RJT represented on Figure 7.4. Algorithm 4 runs in polynomial time since there are $|V|$ iterations, whose individually have a time complexity in the worst case of $|V|$. The following proposition shows that Algorithm 4 produces an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ which is minimal for $\preccurlyeq$, in the sense that it satisfies a converse of (7.5).

Proposition 7.7. *Algorithm 4 produces an RJT such that the root vertex C_v of v is C'_v, satisfying* $\mathrm{offspring}(C_v) = \{v\}$, *that admits $\preccurlyeq$ as a topological order, and such that $(u \in C_v) \Rightarrow (u \preccurlyeq v)$. Moreover, its cluster vertices are minimal in the sense that*

$$u \in C_v \iff \begin{cases} \exists w \in V \ s.t.\ C_v \dashrightarrow C_w \ and \ u \in \mathrm{fa}(w), \\ \qquad\qquad C_u \dashrightarrow C_v. \end{cases} \tag{7.6}$$

Although Proposition 7.7 does not give a guarantee about the width of the RJT built by Algorithm 4, the equivalence (7.6) ensures that given a topological ordering the cluster vertices contain only the required vertices to ensure the running intersection property.

Remark 10. Algorithm 4 takes in input a topological order on G. For a practical use, we recommend to use Algorithm 7 in Appendix B, which builds simultaneously the RJT and a "good" topological order. △

[4] Although the particular RJT obtained by Algorithm 4 is a bucket tree, considering RJTs that are not bucket trees is also useful. Figure 8.7 shows an application where it is interesting to use an RJT that is not a bucket tree.

Proof of Proposition 7.7. Algorithm 4 obviously converges given that it has only a finite number of iterations. If G is not connected, the algorithm is equivalent to its separate application on each of the connected components, which each yield a tree. W.l.o.g., we prove properties of the algorithm under the assumption that G is connected. To simplify notations, we denote C'_v by C_v, and check that it indeed corresponds to the root vertex of v.

We first prove that $\preccurlyeq$ is a topological order on $\mathcal{G}$. First, remark that $(u \in C_v) \Rightarrow (u \preccurlyeq v)$. Indeed, if $u \in C_v$, then either Step 4 of the algorithm ensures that $u \in \mathrm{fa}(v)$ and $u \preccurlyeq v$ or $u \notin \mathrm{fa}(v)$ and there exists x such that $u \in C_x$ and $C_v \to C_x$. But by Step 6 of Algorithm 4, the fact that $C_v \to C_x$ entails that v is the maximal element of $C_x \backslash \{x\}$ for the topological order, so that $u \prec v$. Furthermore, Step 6 ensures that $(C_u, C_v) \in \mathcal{A}$ implies $u \in C_v$. We deduce from the previous result that $(C_u, C_v) \in \mathcal{A}$ implies $u \preccurlyeq v$, and $\preccurlyeq$ is a topological order on $\mathcal{G}$.

Then we show that (7.6) holds. We first show that $(u \in C_v) \Rightarrow C_u \dashrightarrow C_v$ and $u \in C'$ for any C' on path $C_u \dashrightarrow C_v$. Either $u = v$ and this is obvious, or $u \in \mathrm{pa}_{\mathcal{G}}(C_v)$; and by recursion either $C_u \dashrightarrow C_v$ or $u \in C_r$ with C_r the root of the tree which is also the first element in the topological order. But, unless $u = r$, this is excluded given that $u \in C_r$ implies $u \preccurlyeq r$. Note that this shows that C_u is indeed the unique minimal element of the set $\{C\colon\ u \in C\}$ for the partial order defined by the arcs of the tree. To show the first part of (7.6), we just need to note that either $u \in \mathrm{fa}(v)$ and the result holds, or there must exist x such that $C_v \to C_x$ and $u \in C_x$ and by recursion, there exists w such that $C_v \dashrightarrow C_w$ and $u \in \mathrm{fa}(w)$.

Finally, we prove that we have constructed an RJT. Indeed, if two vertices C_v and $C_{v'}$ contain u then since $\mathcal{G}$ is singly connected, the trail connecting C_v and $C_{v'}$ must be composed of vertices on the paths $C_v \dashleftarrow C_u$ and $C_u \dashrightarrow C_{v'}$, and we have shown in the previous paragraph that that u belongs to any C' on $C_v \dashleftarrow C_u$ and $C_u \dashrightarrow C_{v'}$, and so the running intersection property holds. Finally, property (ii) of Definition 6.1 must holds because the fact that C_u is minimal among all cluster vertices containing u together with the running intersection property entails that the cluster vertices containing u are indeed a subtree of $\mathcal{G}$ with root C_u. □

7.5.2 Characterizing the built RJT

Proposition 7.7, in addition to proving the correctness of Algorithm 4, provides the benefit of using Algorithm 4, but it characterizes the content of the cluster vertices based on the topology of the obtained RJT, which is itself produced by the algorithm (note that the composition of cluster vertices depends only on $\preccurlyeq$ via the partial order of the tree). The cluster vertices of any RJT and those produced by Algorithm 4 admit however more technical characterizations using only $\preccurlyeq$ and the information in G, which we present this section. These characterizations will be useful in the proofs of Chapter 9. For each vertex v in V, we introduce

$$T_{\succcurlyeq v} = \{w \in V_{\succcurlyeq v} \mid \text{ there is a } v\text{-}w \text{ trail in } V_{\succcurlyeq v}\}.$$

Proposition 7.8. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be an RJT satisfying* $\mathrm{offspring}(C_v) = \{v\}$, *and $\preccurlyeq$ be a topological* *order on $\mathcal{G}$. Then $\preccurlyeq$ induces a topological order on G and*

$$w \in T_{\succcurlyeq v} \implies C_v \dashrightarrow C_w, \tag{7.7a}$$

$$\left.\begin{array}{r} \mathrm{ch}(u) \cap T_{\succcurlyeq v} \neq \emptyset \\ \text{and } u \preccurlyeq v \end{array}\right\} \implies u \in C_v. \tag{7.7b}$$

Proof of Proposition 7.8. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be an RJT satisfying offspring$(C_v) = \{v\}$, and $\preccurlyeq$ be a topological order on $\mathcal{G}$. Property 1 in Proposition 7.4 ensures that $\preccurlyeq$ induces a topological order on G.

We start by proving (7.7a). Let v and w be vertices such that $w \succ v$ and that there is a v-w trail Q in $V_{\succcurlyeq v}$. Let $s_0, \ldots, s_k$ be the vertices where Q has a v-structure[5] and $t_1, \ldots, t_k$ the vertices with diverging arcs in Q. Note that, since the trail is included in $V_{\succcurlyeq v}$, the first vertices of the trail have to be immediate descendants of v in G so that the trail takes the form $v \dashrightarrow s_0 \dashleftarrow t_1 \dashrightarrow s_1 \ldots t_k \dashrightarrow s_k \dashleftarrow w$, where possibly $s_k = w$ and the last arc is not present. Then, given that $v \prec s_0$, and that $\preccurlyeq$ is topological for $\mathcal{G}$, Proposition 7.4.2 implies that $C_v \dashrightarrow C_{s_0}$. But by the same argument, Property 2 in Proposition 7.4 implies $C_{t_1} \dashrightarrow C_{s_0}$, but since $\mathcal{G}$ is a tree and $v \prec t_1$, we must have $C_v \dashrightarrow C_{t_1} \dashrightarrow C_{s_1}$. By induction on i, we have $C_v \dashrightarrow C_{s_i}$ and thus $C_v \dashrightarrow C_w$, which shows Equation (7.7a).

We now prove (7.7b). Let u and v be two vertices such that $u \preccurlyeq v$ and there is a u-v trail P with $P \backslash \{u\} \subseteq V_{\succcurlyeq v}$. Let w be the vertex right after u on P. We have $u \in \mathrm{fa}(w)$, $w \succcurlyeq v$ and there is a v-w trail in $V_{\succcurlyeq v}$, which implies $C_v \dashrightarrow C_w$ by (7.7a). But, since $u \preccurlyeq v$, the u-v trail is also in $V_{\succcurlyeq u}$, which similarly shows that $C_u \dashrightarrow C_v$. So by (7.5) we have proved (7.7b). □

Proposition 7.9. *The graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ produced by Algorithm 4 is the unique RJT satisfying* offspring$(C_v) = \{v\}$ *such that the topological order $\preccurlyeq$ on G taken as input of Algorithm 4 induces a topological order on $\mathcal{G}$ and the implications in* (7.7) *are equivalences.*

Proof. Note that some visual elements of the proof are given in Figure 7.3. It is sufficient to prove the following inclusions

$$\overline{\mathrm{des}}_{\mathcal{G}}(C_v) \subseteq \{C_w \colon w \in T_{\succcurlyeq v}\}, \tag{7.8a}$$

$$C_v \subseteq \{u \preccurlyeq v \colon \exists w \in T_{\succcurlyeq v}, u \in \mathrm{fa}(w)\}. \tag{7.8b}$$

Indeed, note that by Proposition 7.7, the obtained tree is an RJT so that, by Proposition 7.8, the reverse inequalities hold.

We prove the result by backward induction on (7.8b) and (7.8a). For a leaf C_v of $\mathcal{G}$, $\overline{\mathrm{des}}_{\mathcal{G}}(C_v) = \{C_v\}$ so that (7.8a) holds trivially and $C_v = \mathrm{fa}(v)$ so that (7.8b) holds because $u \in \mathrm{fa}(v)$ implies $u \preccurlyeq v$. Then, assume the induction hypothesis holds for all children of a vertex C_v.

We first show (7.8a) for C_v , i.e. that $(C_v \dashrightarrow C_w) \Rightarrow (w \in T_{\succcurlyeq v})$ (see Figure 7.3). Let C_x be the child of C_v on the path $C_v \dashrightarrow C_w$. By Proposition 7.7, we have $v \prec x$, so that $V_{\succcurlyeq x} \subset V_{\succcurlyeq v}$. Then, using the induction hypothesis, by (7.8b), $(v \in C_x)$ implies that there is a v-x trail in $V_{\succcurlyeq v}$, and

[5]We say that there is a v-structure at v_i on a trail $Q = (v_1, \ldots, v_k)$ if $1 < i < k$ and $v_{i-1} \to v_i \leftarrow v_{i+1}$.

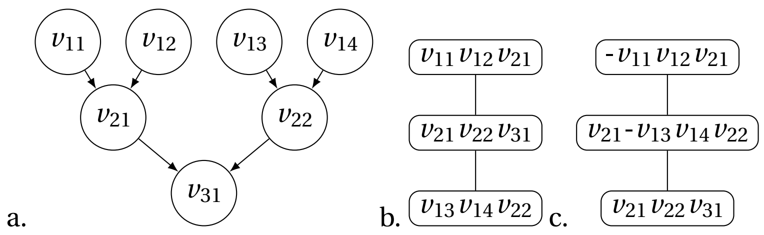


Figure 7.2 – Example of influence diagram whose rooted treewidth is larger than its pathwidth. a. Influence diagram. b. Path decomposition with minimum width. c. RJT with minimum width.

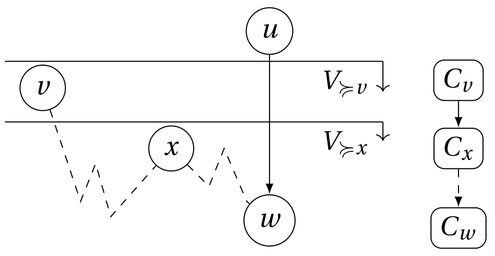


Figure 7.3 – Illustration of the proof of Proposition 7.9. Plain arcs represent arcs, dashed line trails.

by (7.8a), $C_x \dashrightarrow C_w$ implies there is a trail x-w in $V_{\succcurlyeq x}$, so there is a v-w trail in $v \prec x$ in $V_{\succcurlyeq v}$, which shows the result.

We then show (7.8b) for C_v (see Figure 7.3). Indeed if $u \in C_v$, either $u \in \mathrm{fa}(v)$ and u is in the RHS of (7.8b), or there exists a child of C_v, say C_x such that $u \in C_x$ and $u \prec v$, because the algorithm imposes $v = \max_{\preccurlyeq}(C_x \backslash \{x\})$. Since $C_v \dashrightarrow C_x$ there exists a path v-x in $V_{\succcurlyeq v}$, and using induction, by (7.8b), $(u \in C_x)$ implies that $\exists w$ such that $u \in \mathrm{fa}(w)$ and there exists a trail w-x in $T_{\succcurlyeq v}$. But we have shown in Proposition 7.7 that $(v \in C_x) \Rightarrow (v \preccurlyeq x)$, so $T_{\succcurlyeq x} \subset T_{\succcurlyeq v}$ and we have shown that there exists a v-w trail in $T_{\succcurlyeq v}$ with $u \preccurlyeq v$ and $u \in \mathrm{fa}(w)$, which shows the result. □

Finally we conclude this section with Figure 7.2, which shows an example of influence diagram whose rooted-treewidth is equal to 3 and larger than its pathwidth, which is equal to 2. Figure 7.2 also provides the path decomposition of width 2 and the RJT of width 3. This kind of v-shaped graphs are especially challenging for our approach.

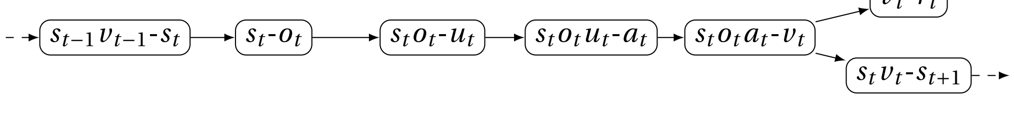


Figure 7.4 – Rooted junction tree produced by Algorithm 4 on the example of Figure 6.2b. The offspring of a vertex is to the right of symbol -.

8 Integer programming on the junction tree polytope

The main goal of this chapter is to prove Theorem 6.1 and Proposition 6.4. It requires to encode the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$ of a strategy $\boldsymbol{\delta}$ using a set of constraints satisfied by a vector of moments $\boldsymbol{\mu}$. In Chapter 7, we introduced some local independences (7.4), which are sufficient to ensure that the probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ of a vector of moments $\boldsymbol{\mu}$ in the marginal polytope of a RJT, factorizes on G. The key aspect in this chapter is to show that given a strategy $\boldsymbol{\delta}$, any vector of moments $\boldsymbol{\mu}$ satisfying the constraints of (6.5) will satisfy (7.4), ensuring that the probability distribution $\mathbb{P}_{\boldsymbol{\mu}}$ factorizes according to the directed acyclic graph. The proof of Theorem 8.1 then follows.

This chapter is organized as follows:

- Section 8.1 gives a proof of Theorem 6.1, which is based on showing that the probability distribution from which derives the feasible vector of moments of MILP (6.6), factorizes according to the influence diagram. Using notation of Chapter 7 we prove that NLP (6.6) models exactly $\mathrm{MEU}(G,\boldsymbol{\rho})$.
- Section 8.2 is dedicated to prove Proposition 6.4. Using notation of Chapter 7, we prove the validity of inequalities (6.7) for MILP (6.6). To the best of our knowledge, such "independence cuts" have not been proposed in combinatorial optimization. We believe that this idea of leveraging conditional independence to obtain valid cuts is fairly general and could be extended to other contexts.
- Section 8.3 details basic knowledge about the McCormick's relaxation. It introduces a dynamic programming like algorithm that computes "good" bounds on the moments of any probabilistic distribution of a PID. Incorporating these new bounds in the McCormick linear inequalities tighten the linear relaxation of MILP (6.6).
- Section 8.4 provides an interpretation of the linear relaxation of MILP (6.6) and the valid inequalities (6.7) in terms of graphs.
- Section 8.5 proves that NLP (6.8) that models $\mathrm{MEU}(G,\boldsymbol{\rho})$. Such NLP is based on the value functions. Using McCormick's linearization technique, this NLP can be turned into an MILP.
- Section 8.6 illustrates the mathematical programs of this chapter and their properties on some simple numerical examples.

8.1 Integer programming using the moments

For convenience, we start by introducing notation that is useful in this section. Then we obtain NLP (6.5) in Section 8.1.2, and then linearize it into MILP (6.6) in Section 8.1.3.

8.1.1 Notation

In this section, we introduce two sets to lighten the writing of the mathematical programs introduced in Section 6.1. Consider an influence diagram $G = (V^{\mathrm{s}}, V^{\mathrm{a}}, A)$. Let $\boldsymbol{\rho} = (\mathcal{X}_V, \mathfrak{p}, \mathbf{r})$ be a parametrization of G such that $\mathcal{X}_V = \prod_{v\in V}\mathcal{X}_v$ the support of the vector of random variables attached to all vertices of G, $\mathfrak{p} = \{p_{v|\mathrm{pa}(v)}\}_{v\in V^{\mathrm{s}}}$ is the collection of fixed and assumed known conditional probabilities, and $\mathbf{r} = \{r_v\}_{v\in V^{\mathrm{r}}}$ is the collection of reward functions[1] $r_v\colon \mathcal{X}_v \to \mathbb{R}$. The reward functions $\mathbf{r}$ will also be viewed as vectors $r_v \in \mathbb{R}^{\mathcal{X}_v}$. Given a gradual RJT $\mathcal{G}$ of G, we introduce the following variant of the marginal polytope $\mathcal{M}^0(\mathcal{G})$ defined in (7.3).

$$\tilde{\mathcal{M}}^0(\mathcal{G}) = \left\{(\mu_{C_v}, \mu_{\check{C}_v})_{v\in V} : (\mu_{C_v})_{v\in V} \in \mathcal{M}^0(\mathcal{G}) \text{ and } \mu_{\check{C}_v} = \sum_{x_v}\mu_{C_v}\right\},$$

where moments $\mu_{\check{C}_v}$ have been introduced. This is for convenience, and all the results could have been written using $\mathcal{M}^0(\mathcal{G})$. By abuse of notation, we say that $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G})$ instead of $(\mu_{C_v}, \mu_{\check{C}_v})_{v\in V} \in \tilde{\mathcal{M}}^0(\mathcal{G})$. We also introduce the polytope

$$\overline{\mathcal{P}}(G,\boldsymbol{\rho}) = \left\{\boldsymbol{\mu} \in \tilde{\mathcal{M}}^0(\mathcal{G}) : \mu_{C_v} = \mu_{\check{C}_v} p_{v|\mathrm{pa}(v)} \text{ for all } v \in V^{\mathrm{s}}\right\}. \tag{8.1}$$

We omit the dependence of $\overline{\mathcal{P}}$ in $\boldsymbol{\rho}$ when it is clear from the context.

8.1.2 An exact Non Linear Program

Using the notation introduced in Section 8.1.1 we write NLP (6.5) more concisely as follows

$$\max_{\boldsymbol{\mu},\boldsymbol{\delta}} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle \tag{8.2a}$$

$$\text{s.t. } \boldsymbol{\mu} \in \overline{\mathcal{P}} \tag{8.2b}$$

$$\boldsymbol{\delta} \in \Delta \tag{8.2c}$$

$$\mu_{C_v} = \delta_{v|\mathrm{pa}(v)}\mu_{\check{C}_v}, \quad \forall v \in V^{\mathrm{a}}. \tag{8.2d}$$

Note that the constraints $\boldsymbol{\delta} \in \Delta$, which are being positive and summing to 1, are implied by the other ones in NLP (8.2) (or (6.5)).[2] We have Theorem 6.1 which we recall here with the notation introduced above:

Theorem 8.1. *Let $(\boldsymbol{\mu},\boldsymbol{\delta})$ be a feasible solution of NLP* (8.2). *Then $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$. Furthermore, $(\boldsymbol{\mu},\boldsymbol{\delta})$ is an optimal solution of NLP* (8.2) *if and only if $\boldsymbol{\delta}$ is an optimal policy of* MEU$(G,\boldsymbol{\rho})$. *In particular, NLP* (8.2) *and* MEU$(G,\boldsymbol{\rho})$ *have the same optimal*

[1] We remind the reader that V^{r} is the set of utility vertices as introduced in Section 6.1.

[2] Indeed, constraint $\boldsymbol{\mu} \in \overline{\mathcal{P}}(G,\mathcal{X},\mathfrak{p},\mathcal{G})$ ensures that, for each $v \in V^{\mathrm{a}}$, μ_{C_v} is a distribution. And constraint $\mu_{C_v} = \delta_{v|\mathrm{pa}(v)}\mu_{\check{C}_v}$ ensures that, if $x_{\mathrm{pa}(v)}$ has non-zero probability according to that distribution, i.e., $\sum_{x_{\check{C}_v\backslash x_{\mathrm{pa}(v)}}} \mu_{\check{C}_v}(x_{\check{C}_v\backslash x_{\mathrm{pa}(v)}}, x_{\mathrm{pa}(v)}) > 0.$, then $\delta_{v|\mathrm{pa}(v)}(\cdot|x_{\mathrm{pa}(v)})$ is a conditional probability.

value.

Theorem 8.1 gives the following corollary:

Corollary 8.2. *Given any gradual RJT $\mathcal{G}$ and any parametrization $\boldsymbol{\rho}$ on G, the set of achievable moments $\mathcal{M}_{\mathcal{G}}(G,\boldsymbol{\rho})$ can be written:*

$$\mathcal{M}_{\mathcal{G}}(G,\boldsymbol{\rho}) = \left\{\boldsymbol{\mu} \in \overline{\mathcal{P}} \colon \exists \boldsymbol{\delta} \in \Delta, \mu_{C_v} = \mu_{\check{C}_v}\delta_{v|\mathrm{pa}_G(v)} \text{ for all } v \text{ in } V^{\mathrm{a}}\right\}.$$

Corollary 8.2 enables to write Problems (6.5) and (8.2) as

$$\max_{\boldsymbol{\mu}\in\mathcal{M}(G)} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle,$$

and it will be useful in the proofs Chapter 9 because it characterizes the set of achievable moments by the set of constraints of (8.2).

Proof of Corollary 8.2. We denote the set $\left\{\boldsymbol{\mu} \in \overline{\mathcal{P}} \colon \exists \boldsymbol{\delta} \in \Delta, \mu_{C_v} = \mu_{\check{C}_v}\delta_{v|\mathrm{pa}_G(v)}, \ \forall v \in V^{\mathrm{a}}\right\}$ by $\mathcal{S}(G)$.

Let $\boldsymbol{\mu} \in \mathcal{S}(G)$. Then, there exists a strategy $\boldsymbol{\delta}$ such that $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a feasible solution of NLP (8.2). Theorem 8.1 ensures that $\boldsymbol{\mu}$ is the vector of moments of the probability distribution $\mathbb{P}_{\boldsymbol{\delta}}$. Therefore, $\boldsymbol{\mu} \in \mathcal{M}(G)$.

Let $\boldsymbol{\mu} \in \mathcal{M}(G)$. By definition, there exists $\boldsymbol{\delta} \in \Delta$ such that $\mu_C(x_C) = \mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C)$ for any $x_C \in \mathcal{X}_C$ and any $C \in \mathcal{V}$. First, Proposition 7.2 ensures that $\boldsymbol{\mu}$ belongs to $\mathcal{M}(G)$. Second, since $\mathbb{P}_{\boldsymbol{\delta}}$ factorizes according to G, Corollary 7.6 ensures that $\mu_{C_v} = \mu_{v|\mathrm{pa}(v)}\mu_{\check{C}_v}$ for any $v \in V$, where $\mu_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) = \mathbb{P}_{\boldsymbol{\delta}}\big(X_v = x_v|X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}\big)$ for any $x_{\mathrm{fa}(v)} \in \mathcal{X}_{\mathrm{fa}(v)}$. By definition of $\mathbb{P}_{\boldsymbol{\delta}}$ in (6.2), we deduce that $\mu_{v|\mathrm{pa}(v)} = \delta_{v|\mathrm{pa}(v)}$ if $v \in V^{\mathrm{a}}$ and $\mu_{v|\mathrm{pa}(v)} = p_{v|\mathrm{pa}(v)}$ otherwise. Therefore, $\boldsymbol{\mu} \in \mathcal{S}_{\mathcal{G}}(G)$. It achieves the proof. □

Proof of Theorems 6.1 and 8.1. Let $(\boldsymbol{\mu},\boldsymbol{\delta})$ be an admissible solution of NLP (8.2). Then $\boldsymbol{\delta}$ is an admissible solution of MEU$(G,\boldsymbol{\rho})$. We now prove that $\boldsymbol{\mu}$ corresponds to the vector of moments of the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$, from which we can deduce that $\mathbb{E}_{\boldsymbol{\delta}}\Big(\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\Big) = \sum_{v\in V^{\mathrm{r}}}\langle r_v, \mu_v\rangle$. The following remark is the a key argument. Let A, P, and D be three disjoint subsets of V, $\mathbb{P}$ a distribution on $\mathcal{X}_V$, $\mu_{A\cup P\cup D}$ the distribution induced by $\mathbb{P}$ on $\mathcal{X}_{A\cup P\cup D}$, and $p_{D|P}$ the conditional distribution of D given P. If $\mu_{A\cup P\cup D} = \mu_{A\cup P}p_{D|P}$, then

$$(X_D \perp\!\!\!\perp X_A \mid X_P)_{\mathbb{P}}. \tag{8.3}$$

By (8.3), we have that the vector $\boldsymbol{\mu}$ satisfies the conditions of Theorem 7.3, and hence it derives from a distribution $\mathbb{P}_{\boldsymbol{\mu}}$ that factorizes on G. Furthermore, constraints in the definition (8.1) of $\overline{\mathcal{P}}$ ensure that $\mathbb{P}_{\boldsymbol{\mu}}(X_v|X_{\mathrm{pa}(v)}) = p_{v|\mathrm{pa}(v)}$ for all $v \in V^{\mathrm{s}}$, which yields the result.

Conversely, let $\boldsymbol{\delta}$ be an admissible solution of MEU$(G,\boldsymbol{\rho})$, and $\boldsymbol{\mu}$ be the vector of moments induced by $\mathbb{P}_{\boldsymbol{\delta}}$. We have $\mu_{C_v} = \mu_{\check{C}_v}p_{v|\mathrm{pa}(v)}$ for v in V^{s} and $\mu_{C_v} = \mu_{\check{C}_v}\delta_{v|\mathrm{pa}(v)}$ for v in V^{a}, and $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a solution of (8.2). Furthermore, $\mathbb{E}_{\boldsymbol{\delta}}\Big(\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\Big) = \sum_{v\in V^{\mathrm{r}}}\langle r_v, \mu_v\rangle$, and (8.2) is equivalent to MEU$(G,\boldsymbol{\rho})$. □

8.1.3 MILP formulation

Now we detail how we turn NLP 8.2 (or equivalently NLP (6.5)) into MILP (6.6). We recall that there always exists at least one optimal strategy which is deterministic (and therefore integral) for $\mathrm{MEU}(G,\boldsymbol{\rho})$. Therefore, we can add integrality constraint (6.3) to (8.2). With this integrality constraint, Equation (8.2d) becomes a *logical constraint*, i.e., a constraint of the form $\lambda y = z$ with λ binary and continuous y and z. Note that constraints can be handled by modern MILP solvers such as `CPLEX` or `Gurobi`, that can solve NLP (8.2) with integrality constraints (6.3). Alternatively, by a classical result in integer programming, we can turn NLP (8.2) into an equivalent MILP by replacing constraint (8.2d) by its McCormick relaxation [100]. For a given $\mathfrak{p}$, let $\mathbf{b} = \Big(b_{\check{C}_v}(x_{\check{C}_v})\Big)_{x_{\check{C}_v}\in\mathcal{X}_{\check{C}_v}, v\in V^{\mathrm{a}}}$ be a vector of upper bounds satisfying

$$\mathbb{P}_{\boldsymbol{\delta}'}\big(X_{\check{C}_v} = x_{\check{C}_v}\big) \leqslant b_{\check{C}_v}(x_{\check{C}_v}) \qquad \forall \boldsymbol{\delta}' \in \Delta, \quad \forall v \in V^{\mathrm{a}}, \quad \forall x_{\check{C}_v} \in \mathcal{X}_{\check{C}_v}. \tag{8.4}$$

For such a vector $\mathbf{b}$, we say that, for a given vertex v, $(\mu_{C_v}, \delta_{v|\mathrm{pa}(v)})$ satisfies McCormick's inequalities (see Section 8.3) if

$$\begin{cases} \mu_{C_v} \geqslant \mu_{\check{C}_v} + (\delta_{v|\mathrm{pa}(v)} - 1)\, b_{\check{C}_v}, \\ \mu_{C_v} \leqslant \delta_{v|\mathrm{pa}(v)}\, b_{\check{C}_v}, \\ \mu_{C_v} \leqslant \mu_{\check{C}_v}. \end{cases} \tag{McCormick(v,b)}$$

Note that the last inequality $\mu_{C_v} \leqslant \mu_{\check{C}_v}$ can be omitted in our case as it is implied by the marginalization constraint $\mu_{\check{C}_v} = \sum_{x_v} \mu_{C_v}$ in the definition of $\tilde{\mathcal{M}}^0(\mathcal{G})$. Given the upper bounds provided by $\mathbf{b}$, we introduce the polytope of valid moments and decisions satisfying all McCormick constraints:

$$\mathcal{Q}^{\mathbf{b}}(G,\mathcal{X}_V,\mathfrak{p},\mathcal{G}) = \Big\{(\boldsymbol{\mu},\boldsymbol{\delta}) \in \tilde{\mathcal{M}}^0(\mathcal{G}) \times \Delta\colon \text{McCormick}(v,b) \text{ is satisfied for all } v \in V^{\mathrm{a}}\Big\}. \tag{8.5}$$

For convenience, we write $\mathcal{Q}^{\mathbf{b}}$ when $(\mathcal{X}_V,\mathfrak{p},\mathcal{G})$ is clear from the context. Thus by using McCormick on Constraints (8.2d), we get that $\mathrm{MEU}(G,\boldsymbol{\rho})$ is equivalent to the following MILP, which is a rewriting of MILP (6.6):

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\delta}} \quad & \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle \\ \text{s.t.} \quad & \boldsymbol{\mu} \in \overline{\mathcal{P}} \\ & \boldsymbol{\delta} \in \Delta^{\mathrm{d}} \\ & (\boldsymbol{\mu},\boldsymbol{\delta}) \in \mathcal{Q}^{\mathbf{b}} \qquad . \end{aligned} \tag{8.6}$$

Remark 11. The strength of the McCormick constraints (McCormick(v,b)) depends on the quality of the bounds $b_{\check{C}_v}$ on $\mu_{\check{C}_v}$. As for a solution $\boldsymbol{\mu}$ of MILP (8.6), $\mu_{\check{C}_v}$ corresponds to a probability distribution, the simplest admissible bound over $\mu_{\check{C}_v}$ is just $\mathbf{b} = \mathbf{1}$, leading to the polytope $\mathcal{Q}^1$. Unfortunately, McCormick's constraints are loose in this case: we show in Section 8.3 that, for any $\boldsymbol{\mu}$ in $\overline{\mathcal{P}}$, there exists $\boldsymbol{\delta}$ in Δ such that $(\boldsymbol{\mu},\boldsymbol{\delta})$ satisfies those McCormick constraints. Hence, when $\mathbf{b} = \mathbf{1}$, McCormick constraints fail to retain any information about the conditional independence statements encoded in the associated nonlinear constraints. Since $\boldsymbol{\delta}$ does not appear outside of the McCormick constraints, their sole interest in that case is to enable the branching decisions on $\boldsymbol{\delta}$ to have an impact on $\boldsymbol{\mu}$. Section 8.3.2 gives an example showing that McCormick constraints do retain information about the conditional independence if bounds $b_{\check{C}_v}$ smaller than 1 are used. Finally, Section 8.3.3 provides a dynamic programming algorithm that efficiently computes such a $\mathbf{b}$. △

8.2 Valid cuts

Classical techniques in integer programming, such as branch-and-bound algorithms, rely on solving the relaxation of the MILP to obtain a lower bound on the value of the objective. For MILP (8.6) the relaxation is likely to be poor, and so the MILP is not well solved by off-the-shelf solvers. Indeed as explained in Remark 11, when $\mathbf{b} = \mathbf{1}$, the McCormick inequalities completely fail to capture, in the linear relaxation of the MILP (8.6), the conditional independence encoded by the nonlinear constraints (8.2d). Further, improving the bound $\mathbf{b}$ does not completely address the issue. In this section, our goal is to prove Proposition 6.4. We introduce the inequalities (6.7) and we prove their validity for MILP (8.6). It strengthens the relaxation and ease the MILP resolution in practice (see Section 8.6). Recall that a *valid cut* for an MILP is an (in)equality that is satisfied by any solution of the MILP, but not necessarily by solutions of its linear relaxation. A family of valid cuts is stronger than another when the former yields a polytope strictly included in the latter.

8.2.1 Constructing valid cuts

By restricting ourselves to vectors of moments $\boldsymbol{\mu} \in \overline{\mathcal{P}}$, we have imposed

$$\mathbb{P}_{\boldsymbol{\mu}}(X_v | X_{\check{C}_v}) = p_{v|\mathrm{pa}(v)} \quad \text{for all } v \text{ in } V^{\mathrm{s}},$$

because $\boldsymbol{\mu} \in \overline{\mathcal{P}}$ must satisfy $\mu_{C_v} = \mu_{\check{C}_v} p_{v|\mathrm{pa}(v)}$. It turns out that in the linear relaxation of MILP (8.6), the equality $\mu_{C_v} = \mu_{\check{C}_v} \delta_{v|\mathrm{pa}(v)}$ for $v \in V^{\mathrm{a}}$ does not longer hold in general because the McCormick's relaxation is not exact. Hence, in the linear relaxation of MILP (8.6), we do not longer have $\mathbb{P}_{\boldsymbol{\mu}}(X_v | X_{\mathrm{pa}(v)}) = \mathbb{P}_{\boldsymbol{\delta}}(X_v | X_{\mathrm{pa}(v)}) = \delta_{v|\mathrm{pa}(v)}$ for $v \in V^{\mathrm{a}}$.

A key question is therefore whether we can enforce some conditional probabilities implied by the nonlinear constraints, and thus lost in the linear relaxation of MILP (8.6), through linear constraints. This seems possible because, as an indirect consequence of setting the conditional distributions $p_{v|\mathrm{pa}(v)}$ for $v \in V^{\mathrm{s}}$, there are other conditional probability distributions whose value does not depend on $\boldsymbol{\delta}$. We characterize such conditional probability distributions using the notion of strategy independent set introduced in Chapter 6, [3] which we recall here: Let C be a subset of vertices in V. A set of variables X_D such that $D \subseteq C$ is strategy independent set in C if it satisfies the following property:

For all parametrization $\boldsymbol{\rho}$ such that $(G, \boldsymbol{\rho})$ is a PID, $\mathbb{P}_{\boldsymbol{\delta}}(X_D | X_{C \backslash D})$ does not depend on $\boldsymbol{\delta}$.

For such a subset D, the following proposition ensures that we can add linear constraints in

[3]This type of property is well known in the literature on causality in graphical models, where the policies are viewed as *interventions* and some conditional probabilities are shown to be *invariant* under interventions; see, e.g., Koller and Friedman [75, Definition 21.3, p. 1019] or Peters et al. [119, Remark 6.40, p. 113].

MILP (8.6).

Proposition 8.3. *Let C be a subset of vertices and D a strategy independent set in C. The following equalities are valid for MILP* (8.6) *and are linear in $\boldsymbol{\mu}$:*

$$\mu_C = \mu_{C\backslash D}\, p_{D|C\backslash D}. \tag{8.7}$$

Proof. Let $(\boldsymbol{\mu}, \boldsymbol{\delta})$ be a feasible solution of MILP (8.6). Theorem 8.1 ensures that there exists $\boldsymbol{\delta} \in \Delta^{\mathrm{d}}$ such that $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}}$. Hence, it follows that

$$\begin{aligned}\mu_C(x_C) = \mathbb{P}_{\boldsymbol{\delta}}\big(X_C = x_C\big) &= \mathbb{P}_{\boldsymbol{\delta}}\big(X_D = x_D | X_{C\backslash D} = x_{C\backslash D}\big)\mathbb{P}_{\boldsymbol{\delta}}\big(X_{C\backslash D} = x_{C\backslash D}\big)\\ &= p_{D|C\backslash D}\big(x_D | x_{C\backslash D}\big)\mu_{C\backslash D}(x_{C\backslash D})\end{aligned}$$

These equalities are linear because $p_{D|C\backslash D}$ does not depend on $\boldsymbol{\delta}$. $\square$

Lemma 6.2, which will be proved in Section 8.2.2, ensures that the largest inclusion-wise strategy independent set $C^{\perp\!\!\!\perp}$ in C exists and is unique. Then, Proposition 6.4 follows immediately from Proposition 8.3 by setting $D = C^{\perp\!\!\!\perp}$. In addition, we will give a full characterization of $C^{\perp\!\!\!\perp}$ in Section 8.2.2. Such a characterization is key in proving Proposition 6.3, ensuring that $C^{\perp\!\!\!\perp}$ can be computed in $O(|C|(|V| + |A|))$. We believe that this complexity can be improved. To compute $(C_v^{\perp\!\!\!\perp})_{v\in V^{\mathrm{a}}}$, the total complexity is in the worst case in $O(|V^{\mathrm{a}}|w(\mathcal{G})(|V| + |A|))$. In the experiments of Section 8.6, the time to compute all the $C^{\perp\!\!\!\perp}$ and all $p_{C^{\perp\!\!\!\perp}|C^{\not\perp\!\!\!\perp}}$ was negligible compared to the time needed to solve an LP or the MILP.

Finding the largest inclusion-wise strategy independent set is motivated by the fact that the large D the stronger the valid cuts (8.7) in the following sense: Given a strategy independent set D in C, for any set $D' \subseteq D$, the equalities $\mu_C = \mu_{C\backslash D'}\, p_{D'|C\backslash D'}$ are valid for MILP (6.6) and are linear in $\boldsymbol{\mu}$. This fact comes from the following stability property of the strategy independent sets.

Lemma 8.4. *Let D be a strategy independent set in C, and consider $D' \subseteq D$. Then, D' is a strategy independent set in C.*

Proof. It suffices to prove that $\mathbb{P}_{\boldsymbol{\delta}}\big(X_{D'} = x_{D'} | X_{C\backslash D'} = x_{C\backslash D'}\big)$ does not depend on $\boldsymbol{\delta}$. We compute the conditional probability using Bayes law:

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\delta}}\big(X_{D'} = x_{D'} | X_{C\backslash D'} = x_{C\backslash D'}\big) &= \frac{\mathbb{P}_{\boldsymbol{\delta}}\big(X_D = x_D | X_{C\backslash D} = x_{C\backslash D}\big)}{\mathbb{P}_{\boldsymbol{\delta}}\big(X_{D\backslash D'} = x_{D\backslash D'} | X_{C\backslash D} = x_{C\backslash D}\big)}\\ &= \frac{\mathbb{P}_{\boldsymbol{\delta}}\big(X_D = x_D | X_{C\backslash D} = x_{C\backslash D}\big)}{\sum_{x''_{D'}} \mathbb{P}_{\boldsymbol{\delta}}\big(X_{D\backslash D'} = x_{D\backslash D'}, X_{D'} = x''_{D'} | X_{C\backslash D} = x_{C\backslash D}\big)}\end{aligned}$$

The numerator and the denominator in the last fraction above do not depend on $\boldsymbol{\delta}$ because D is strategy independent set in C, which achieves the proof. $\square$

Consider a PID $(G, \boldsymbol{\rho})$ with $G = (V^{\mathrm{s}}, V^{\mathrm{a}}, A)$ and $\boldsymbol{\rho} = (\mathcal{X}_V, \mathfrak{p}, \mathbf{r})$. Let $\mathcal{G}$ be a gradual RJT on G. We

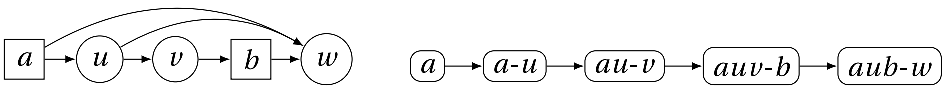


Figure 8.1 – influence diagram and its RJT with a non valid cut (6.7) for $C = \{a, u, v, b\}$ with $C^{\perp\!\!\perp} = \{u\}$.

define $\mathcal{P}^{\perp\!\!\perp}$ as the polytope we obtain when we strengthen $\overline{\mathcal{P}}$ with the valid cuts:

$$\mathcal{P}^{\perp\!\!\perp}(G, \mathcal{X}_V, \mathfrak{p}, \mathcal{G}) = \left\{ \boldsymbol{\mu} \in \overline{\mathcal{P}} \colon \mu_{C_v} = p_{C_v^{\perp\!\!\perp}|C_v^{\not\perp\!\!\perp}} \sum_{x_{C_v^{\perp\!\!\perp}}} \mu_{C_v} \text{ for all } v \in V^{\mathrm{a}} \right\}. \tag{8.8}$$

Given a vector $\mathbf{b}$ satisfying (8.4) we introduce the following strengthened version of the MILP (8.6).

$$\max_{\boldsymbol{\mu},\boldsymbol{\delta}} \sum_{v \in V^{\mathrm{r}}} \langle r_v, \mu_v \rangle \text{ subject to } \boldsymbol{\mu} \in \mathcal{P}^{\perp\!\!\perp}, \ \boldsymbol{\delta} \in \Delta^{\mathrm{d}}, \ (\boldsymbol{\mu}, \boldsymbol{\delta}) \in \mathcal{Q}^{\mathbf{b}}. \tag{8.9}$$

Figure 8.1 provides an example of an influence diagram where the introduction of valid cut of the form (6.7) reduces the size of the initial polytope. Indeed the cluster $C = \{a, u, v, b\}$ leads to $C^{\perp\!\!\perp} = \{u\}$, and the resulting cut (6.7) is not implied by the linear inequalities of (8.6). Indeed, suppose that $\mathcal{X}_a = \mathcal{X}_v = \{0\}$, while $\mathcal{X}_u = \mathcal{X}_b = \{0, 1\}$. Then the solution defined by $\mu_{auvb}(0, i, 0, i) = 0.5$ and $\mu_{auvb}(0, i, 0, 1 - i) = 0$ for $i \in \{0, 1\}$ is in the linear relaxation of MILP (8.6) but does not satisfy (6.7). To compute $C^{\perp\!\!\perp}$, we have used the characterization provided in Section 8.2.2.

Remark 12. In the definition of $\mathcal{P}^{\perp\!\!\perp}$, we decided to introduce valid cuts of the form (8.7) only for sets of vertices C of the form C_v with $v \in V^{\mathrm{a}}$. This is to strike a balance between the number of constraints added and the number of independences enforced. This choice is however heuristic, and it could notably be relevant to introduce constraints of the form (6.7) for well chosen $C \subsetneq C_v$. △

8.2.2 Characterization of $C^{\perp\!\!\perp}$

In this section, we show the existence of $C^{\perp\!\!\perp}$ by giving its closed form. In order to characterize $C^{\perp\!\!\perp}$, we need some concepts from graphical model theory. The first notions make it possible to identify conditional independence from properties of the graph. Let $D \subseteq V$ be a subset of vertices. Let P be a trail $v_1 \rightleftharpoons \cdots \rightleftharpoons v_n$. We say that v_i is a *v-structure* on P if $1 < i < n$ and the subtrail $v_{i-1} \rightleftharpoons v_i \rightleftharpoons v_{i+1}$ is of the form $v_{i-1} \to v_i \leftarrow v_{i+1}$. A trail $v_1 \rightleftharpoons \cdots \rightleftharpoons v_n$ is *active given D* if, for all v-structure $v_{i-1} \to v_i \leftarrow v_{i+1}$ on P, v_i or one of its descendant is in D, and any vertex of the trail that is not a v-structure is not in D. Two sets of vertices B_1 and B_2 are said to be *d-separated* by D in G, and we will denote this property by $B_1 \perp B_2 \mid D$, if there is no active trail between B_1 and B_2 given D. We have $X_{B_1} \perp\!\!\perp X_{B_2} \mid X_D$ for any probability distribution factorizing on G if and only if B_1 and B_2 are d-separated by D [75, Theorem 3.4].

The other notion we need is the one of *augmented model* [75, Chapter 21]. Informally, the idea behind augmented models is as follows: the strategies specify some conditional distributions in the model; the statement that some conditional probabilities *do not depend* on the strategy is a priori to be understood for a *functional notion of independence* which is different from *probabilistic independence*. However, and somewhat surprisingly, by considering a randomized version of the strategy (and thus of the corresponding conditional distribution) the two notions of independence actually coincide. This motivates to "augment" the graph G with additional vertices associated with policies themselves, viewed as random variables, and which are each a parent of the vertex whose conditional distribution they define. The fact that the set of introduced policy variables are conditionally independent of a set of variables X_D given $X_{C\backslash D}$ in the augmented graph turns out to be equivalent to the fact that the value $\mathbb{P}_{\boldsymbol{\delta}}(X_D|X_{C\backslash D})$ does not depend on the choice of $\boldsymbol{\delta}$.

Formally, consider $(G,\boldsymbol{\rho})$, a PID with $G = (V^{\mathrm{s}}, V^{\mathrm{a}}, A)$, and let $V = V^{\mathrm{a}} \cup V^{\mathrm{s}}$. For each $v \in V^{\mathrm{a}}$, we introduce a vertex ϑ_v and a corresponding random variable θ_v. The variable θ_v takes its value in the space Δ_v of conditional distributions on X_v given $X_{\mathrm{pa}(v)}$. Let $G^\dagger$ be the digraph with vertex set $V_{G^\dagger} = V \cup \vartheta_{V^{\mathrm{a}}}$, where $\vartheta_{V^{\mathrm{a}}} = \{\vartheta_v\}_{v\in V^{\mathrm{a}}}$, and arc set $A_{G^\dagger} = A \cup \{(\vartheta_v, v), \forall v \in V^{\mathrm{a}}\}$. Such a graph $G^\dagger$, called augmented graph, is illustrated on Figure 8.2, where vertices in $G^\dagger\backslash G$ are represented as rectangles with rounded corners. The *augmented model* of $(G,\boldsymbol{\rho})$ is the collection of probability distributions factorizing on $G^\dagger$ such that $\mathcal{X}_v$ is defined as in $\boldsymbol{\rho}$ for each v in V, $\mathcal{X}_{\vartheta_v} = \Delta_v$, and

$$\mathbb{P}_{G^\dagger}\Big(X_v = x_v | X_{\mathrm{pa}_{G^\dagger}(v)} = x_{\mathrm{pa}_{G^\dagger}(v)}\Big) = \begin{cases} \theta_v^o(x_v|x_{\mathrm{pa}(v)}) & \text{if } v \in V^{\mathrm{a}}, \\ p_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) & \text{if } v \in V^{\mathrm{s}}, \end{cases}$$

where $x_{\mathrm{pa}_{G^\dagger}(v)} = (x_{\mathrm{pa}_G(v)}, \theta_v^o)$ for $v \in V^{\mathrm{a}}$, and $x_{\mathrm{pa}_{G^\dagger}(v)} = x_{\mathrm{pa}_G(v)}$ for $v \in V^{\mathrm{s}}$.

A probability distribution of the augmented model is specified by choosing the marginal distributions of the θ_v, for $v \in V^{\mathrm{a}}$. In the rest of the thesis, we denote by $\mathbb{P}_{G^\dagger}$ the distribution of the augmented model with uniformly distributed $\theta_v(\cdot|x_{\mathrm{pa}(v)})$ for each v in V^{a}, and each value of $x_{\mathrm{pa}_G(v)} \in \mathcal{X}_{\mathrm{pa}_G(v)}$.

With these definitions, a strategy $\boldsymbol{\delta}$ can now be interpreted as a value taken by $\theta_{V^{\mathrm{a}}}$, and we have

$$\mathbb{P}_{\boldsymbol{\delta}}(X_D = x_D | X_M = x_M) = \mathbb{P}_{G^\dagger}(X_D = x_D | X_M = x_M, \theta_{V^{\mathrm{a}}} = \boldsymbol{\delta}), \tag{8.10}$$

for any set $D, M \subseteq V$. Note that in general $\mathbb{P}_{G^\dagger}(X_D = x_D|X_M = x_M)$ is the expected value over $\theta_{V^{\mathrm{a}}}$ of $\mathbb{P}_{\theta_{V^{\mathrm{a}}}}(X_D = x_D|X_M = x_M)$. The following result, which is an immediate consequence of (8.10), characterizes the pairs (D,M) such that for any parametrization $\boldsymbol{\rho}$ the conditional probability distribution $\mathbb{P}_{\boldsymbol{\delta}}(X_D|X_M)$ does not depend on strategy $\boldsymbol{\delta}$.

Proposition 8.5. *We have $\mathbb{P}_{\boldsymbol{\delta}}(X_D|X_M) = \mathbb{P}_{G^\dagger}(X_D|X_M)$ for any parametrization $\boldsymbol{\rho}$ on G, any strategy $\boldsymbol{\delta}$, and any M such that $\mathbb{P}_{\boldsymbol{\delta}}(X_M) > 0$ if and only if D is d-separated from $\vartheta_{V^{\mathrm{a}}}$ given M in $G^\dagger$.*

Note that this is a particular case of a result known in the causality theory for graphical models [see, e.g. 75, Proposition 21.3]. The following corollary gives a new characterization of the strategy independent sets:

Corollary 8.6. *Given a set C, a subset D of C is strategy independent in C if and only if D is d-separated from $\vartheta_{V^{\mathrm{a}}}$ given $C\backslash D$ in $G^\dagger$.*

Proof. If D is a strategy independent set in C. Then, the property (8.10) ensures that for any parametrization $\boldsymbol{\rho}$, $\mathbb{P}_{G^\dagger}(X_D = x_D | X_M = x_M, \theta_{V^{\mathrm{a}}} = \boldsymbol{\delta})$ does not depend on $\boldsymbol{\delta}$. Hence, for any parametrization $\boldsymbol{\rho}$ and any strategy $\boldsymbol{\delta}$ we obtain $\mathbb{P}_{\boldsymbol{\delta}}(X_D | X_{C\backslash D}) = \mathbb{P}_{G^\dagger}(X_D | X_{C\backslash D})$. Proposition 8.5 ensures that D is d-separated from $\vartheta_{V^{\mathrm{a}}}$ given $C\backslash D$ in $G^\dagger$.

If D is d-separated from $\vartheta_{V^{\mathrm{a}}}$ given M in $G^\dagger$. Then, Koller and Friedman [75, Theorem 3.3] ensures that for any probability distribution $\mathbb{P}$ factorizing on $G^\dagger$, we have $(X_D \perp\!\!\!\perp \vartheta_{V^{\mathrm{a}}} | X_{C\backslash D})_{\mathbb{P}}$. Hence, we obtain $\mathbb{P}_{G^\dagger}(X_D | X_{C\backslash D}, \theta_{V^{\mathrm{a}}}) = \mathbb{P}_{G^\dagger}(X_D | X_{C\backslash D})$ for any probability distribution factorizing on $G^\dagger$. Using (8.10), we get that $\mathbb{P}_{\boldsymbol{\delta}}(X_D | X_{C\backslash D}) = \mathbb{P}_{G^\dagger}(X_D | X_{C\backslash D})$ for any parametrization $\boldsymbol{\rho}$ and any strategy $\boldsymbol{\delta}$, which concludes the proof. □

The characterization of Corollary 8.6 allows us to define the strategy independent sets in C using the notion of d-separation, which is a pure graphical property. Now we can prove Lemma 6.2 which we recall here:

Lemma 6.2. *Let $D, D' \subseteq C$. If D and D' are strategy independent in C, then $D \cup D'$ is strategy independent in C.*

Proof. It suffices to prove that $\vartheta_{V^{\mathrm{a}}} \not\perp D \cup D' | C\backslash(D \cup D')$. Suppose that it is not true, i.e., there exists an active trail between $\vartheta_{V^{\mathrm{a}}}$ and $D \cup D'$ that is active given $C\backslash(D \cup D')$. Let P be such a trail. Hence, there exist two vertices $u, v \in V$ such that $u \in D \cup D'$ and $v \in V^{\mathrm{a}}$, and P is active between u and ϑ_v given $C\backslash(D \cup D')$. Without loss of generality, we suppose that P is minimal in the sense that $P \cap (D \cup D') = \{u\}$. Indeed, otherwise we consider the nearest vertex of P that belongs to $D \cup D'$ and the trail from this vertex to ϑ_v is also active given $C\backslash(C \cup C')$. Suppose that $u \in D$. Since D is a strategy independent set in C, we have $\vartheta_v \not\perp u | C\backslash D$. Hence, the trail P is not active given $C\backslash D$. Therefore, either there is a v-structure of P with no descendant in $C\backslash D$ or there is a vertex on P that is not a v-structure and that belongs to $C\backslash D$. Since P is active given $C\backslash(D \cup D')$, it means that in all the v-structures of P, there is at least one descendant that is in $C\backslash(D \cup D') \subseteq C\backslash D$. Therefore, all the v-structures are active given $C\backslash D$.

We deduce that there is a vertex on P that is not a v-structure and that belongs to $C\backslash D$. Since P is active given $C\backslash(D \cup D')$, we have $x \in D'\backslash D$. It contradicts the fact that P is minimal. □

Given a set C, the stability property of Lemma 6.2 enables to define $C^{\perp\!\!\perp}$ as the largest inclusion-wise strategy independent set in C. In fact, we can give a full characterization of $C^{\perp\!\!\perp}$ as stated in the following theorem.

Theorem 8.7. *$C^{\perp\!\!\perp}$ is equal to $\Big\{v \in C \colon v \perp \vartheta_{V^{\mathrm{a}}} \,|\, C\backslash\{v\}\Big\}$.*

With this characterization, the reader can check the value of $C^{\perp\!\!\perp}$ on the example of Figure 8.1.

If we want to use the valid cuts in (8.8) in practice, we must compute $C^{\perp\!\!\perp}$ and $p_{C^{\perp\!\!\perp}|C^{\not\perp\!\!\perp}}$ efficiently. Theorem 8.7 ensures that $C^{\perp\!\!\perp}$ is easy to compute using any d-separation algorithm (and more efficient algorithms are presumably possible), and Proposition 8.5 ensures that, if we solve the inference problem on the RJT for an arbitrary strategy, *e.g.,* one where decisions are made with uniform probability, we can deduce $p_{C^{\perp\!\!\perp}|C^{\not\perp\!\!\perp}}$ from the distribution μ_C obtained. Theorem 8.7 is an immediate corollary of the following lemma.

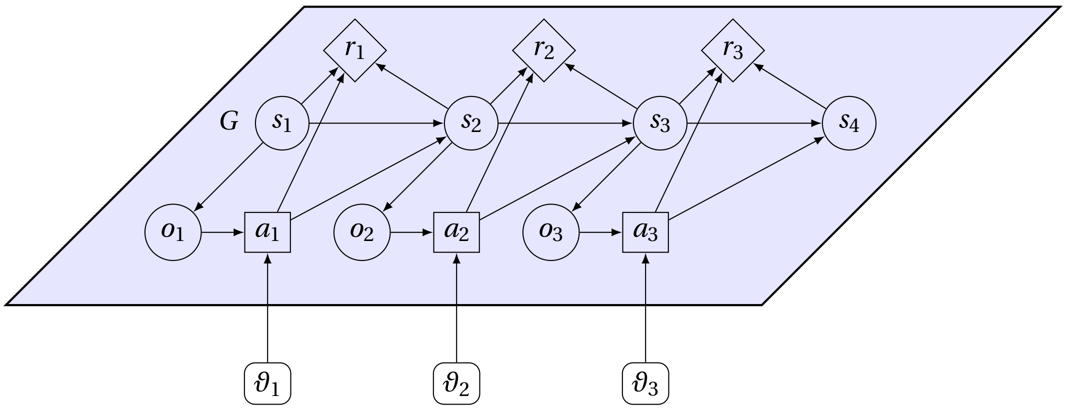


Figure 8.2 – Example of augmented graph $G^\dagger$ on a POMDP.

Lemma 8.8. *Let B and C be two sets of vertices. Then the smallest inclusion-wise subset $M \subseteq C$ such that*

$$B \perp \big(C \backslash (B \cup M)\big) \mid M \tag{8.11}$$

is $M^* := \Big\{v \in C \backslash B \colon v \not\perp B \mid C \backslash (B \cup \{v\})\Big\}$. Furthermore, a set $M \subseteq C$ satisfies (8.11) *if and only if $M^* \subseteq M$.*

Lemma 8.8 has been recently proven by two of the authors [27, Theorem 1]. Using their terminology, M^* is the *Markov blanket of B in C*. Note that if $C = V$ this is the usual Markov blanket.

Proof of Theorem 8.7. We first prove that $C^{\not\perp\!\!\!\perp} = C \backslash C^{\perp\!\!\!\perp}$ corresponds to the Markov Blanket of ϑ_V^{a} in C. It suffices to show that $\vartheta_V^{\mathrm{a}} \perp C \backslash (B \cup C^{\not\perp\!\!\!\perp}) | C^{\not\perp\!\!\!\perp}$. Since $C^{\perp\!\!\!\perp}$ is strategy independent, Corollary 8.6 ensures that $\vartheta_V^{\mathrm{a}} \perp C^{\perp\!\!\!\perp} | C^{\not\perp\!\!\!\perp}$ which is equivalent to $\vartheta_V^{\mathrm{a}} \perp C \backslash C^{\not\perp\!\!\!\perp} | C^{\not\perp\!\!\!\perp}$. Since $\vartheta_V^{\mathrm{a}} \cap C = \emptyset$, we obtain

$$\vartheta_V^{\mathrm{a}} \perp C \backslash \big(\vartheta \cup C^{\not\perp\!\!\!\perp}\big) | C^{\not\perp\!\!\!\perp}.$$

It ensures that $C^{\not\perp\!\!\!\perp}$ satisfies (8.11). By definition of $C^{\perp\!\!\!\perp}$, $C^{\not\perp\!\!\!\perp}$ is the smallest inclusion-wise subset of C satisfying (8.11). Therefore, Lemma 8.8 ensures that $C^{\not\perp\!\!\!\perp}$ is the Markov Blanket of ϑ_V^{a} in C and can be written $C^{\not\perp\!\!\!\perp} = \big\{v \in C \colon v \not\perp \vartheta_V^{\mathrm{a}} \mid C \backslash \{v\}\big\}$. We conclude that $C^{\perp\!\!\!\perp} = C \backslash C^{\not\perp\!\!\!\perp} = \big\{v \in C \colon v \perp \vartheta_V^{\mathrm{a}} \mid C \backslash \{v\}\big\}$. □

The characterization of Theorem 8.7 gives a closed form of $C^{\perp\!\!\!\perp}$, which helps to compute $C^{\perp\!\!\!\perp}$. In addition, it is key in proving Proposition 6.3 which we recall here:

Proposition 6.3. *Given a set C in V, $C^{\perp\!\!\!\perp}$ can be computed in $O\big(|C|(|V| + |A|)\big)$.*

Proof. Theorem 8.7 ensures that computing $C^{\perp\!\!\!\perp}$ requires to find all vertices v in C that are d-separated from ϑ_V^{a} given $C \backslash \{v\}$. Given a vertex $v \in C$, computing the set $\{u \in C \colon \vartheta_V^{\mathrm{a}} \perp u | C \backslash \{v\}$ can be done in $O(|V| + |A|)$ using the *Bayes-Ball algorithm* [138, Theorem 4]. By running the algorithm for each vertex v in C, the total complexity is in $O(|C|(|V| + |A|))$, which achieves the proof. □

8.3 McCormick Relaxation

McCormick inequalities allow to turn NLP (8.2) into MILP (8.6). Further good bounds **b** on the vector of moments ease the resolution of MILP (8.6). In this section we first discuss these relaxation, show that in the NLP loose bounds are useless while tight bounds improve the MILP formulation. Finally, we give an algorithm to compute good quality bounds.

8.3.1 Review of McCormick's relaxation

For the sake of completeness we briefly recall McCormick's relaxation, and condition for exactness if all of the variables but one are binary.

Proposition 8.9. *Consider the variables* $(x,y,z)\in[0,1]^3$ *and the following constraint*

$$z = xy \tag{8.12}$$

Further, assume that we have an upper bound $y \leqslant b$*. We call* McCormick(8.12) *the following set of constraints*

$$z \geqslant y + xb - b \tag{8.13a}$$
$$z \leqslant y \tag{8.13b}$$
$$z \leqslant bx \tag{8.13c}$$

If x*,* y *and* z *satisfy Equation* (8.12)*, then they also satisfy Equation* (8.13)*. If* x *is a binary variable (that is* $x \in \{0,1\}$*) and Equation* (8.13) *is satisfied, then so is Equation* (8.12)*.*

Proof. Consider $x \in [0,1]$, $y \in [0,b]$ and $z \in [0,1]$, such that $z = xy$. Noting that $(1-x)(b-y) \geqslant 0$ we obtain Constraint (8.13a). Constraints (8.13b) and (8.13c) are obtained by upper bounding by bounding one variable. Now assume that $x \in \{0,1\}$, $y \in [0,b]$ and $z \in [0,1]$ satisfy Equation (8.13). Then, if $x = 1$, constraints (8.13a) and (8.13b) yield $z = y$. Otherwise, as $z \geqslant 0$, we have $z = 0$ by (8.13c). □

8.3.2 Choice of bounds in McCormick inequalities

Using $b_{\check{C}_v} = 1$ leads to loose constraints

Since $\mu_{\check{C}_v}$ is a probability distribution, 1 is an immediate upper bound on $\mu_{\check{C}_v}$. Let $\mathcal{Q}^1$ be the polytope $\mathcal{Q}^b$ obtained using bounds vector b defined by $b_{\check{C}_v} = 1$ for all v in V^{a}.

Proposition 8.10. *Let* $\boldsymbol{\mu}$ *be in* $\overline{\mathcal{P}}$ *(resp.* $\mathcal{P}^{\perp\!\!\perp}$*). Then there exists* $\boldsymbol{\delta}$ *in* Δ *such that* $(\boldsymbol{\mu},\boldsymbol{\delta})$ *belongs to* $\mathcal{Q}^1$*, and the linear relaxation of MILP* (8.6) *(resp. MILP* (8.9)*) with* $\mathbf{b} = 1$ *is equal to* $\max\limits_{\boldsymbol{\mu}\in\overline{\mathcal{P}}} \sum\limits_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle$ *(resp.* $\max\limits_{\boldsymbol{\mu}\in\mathcal{P}^{\perp\!\!\perp}} \sum\limits_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle$*).*

Proof. We write the proof with $\boldsymbol{\mu} \in \overline{\mathcal{P}}$ and it is exactly the same if $\boldsymbol{\mu} \in \mathcal{P}^{\perp\!\!\perp}$. Let v be a vertex in V^{a}, and let

$$\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) = \begin{cases} \frac{\mu_{\mathrm{fa}(v)}(x_{\mathrm{fa}(v)})}{\mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})} & \text{if } \mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) \neq 0, \\ \mathbb{1}_{e_v}(x_v) & \text{otherwise,} \end{cases}$$

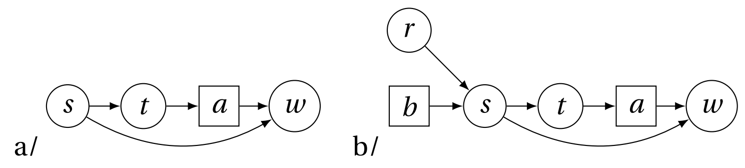


Figure 8.3 – influence diagrams with useful McCormick inequalities.

where e_v is an arbitrary element of $\mathcal{X}_v$. To prove the result, we show that McCormick(v, b) is satisfied for this well-chosen $\delta_{v|\mathrm{pa}(v)}$ and $b_{\check{C}_v} = 1$.

We have

$$\begin{aligned}
\mu_{C_v}(x_{C_v}) - \mu_{\check{C}_v}(x_{\check{C}_v}) &\geqslant \sum_{x'_{C_v\backslash \mathrm{fa}(v)}} \underbrace{\mu_{C_v}(x'_{C_v\backslash \mathrm{fa}(v)}, x_{\mathrm{fa}(v)}) - \mu_{\check{C}_v}(x'_{C_v\backslash \mathrm{fa}(v)}, x_{\mathrm{pa}(v)})}_{\leqslant 0} \\
&= \mu_{\mathrm{fa}(v)}(x_{\mathrm{fa}(v)}) - \mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) \\
&= \frac{1}{\mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})}(\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) - 1) \\
&\geqslant \delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) - 1
\end{aligned}$$

which yields $\mu_{C_v} \geqslant \mu_{\check{C}_v} + (\delta_{v|\mathrm{pa}(v)} - 1)b_{\check{C}_v}$.

Besides, if $\mu_{C_v}(x_{C_v}) \geqslant 0$, following the definition of δ and given that $\mu_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) \leqslant 1$, we have

$$\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}) \geqslant \mu_{\mathrm{fa}(v)}(x_{\mathrm{fa}(v)}) \geqslant \mu_{C_v}(x_{C_v}),$$

and the constraint $\mu_{C_v} \leqslant \delta_{v|\mathrm{pa}(v)} b_{\check{C}_v}$ is satisfied.

Finally, $\mu_{C_v} \leqslant \mu_{\check{C}_v}$ follows from the marginalization constraint $\mu_{\check{C}_v} = \sum_{x_v} \mu_{C_v}$ in the definition of the marginal polytope. □

McCormick inequalities with well-chosen bounds are useful

This section provides examples of influence diagrams where inequalities McCormick(v, b) improves the linear relaxation of MILPs (8.6) and (8.9). Consider the influence diagram on Figure 8.3.a, and assume that we have a bound $\mu_{st} \leqslant b_{st}$. Then, the McCormick relaxation of $\mu_{sta} = \mu_{st}\delta_{a|t}$ reads

$$\begin{cases} \mu_{sta} \geqslant \mu_{st} + b_{st}(\delta_{a|t} - 1) \\ \mu_{sta} \leqslant b_{st}\delta_{a|t} \end{cases}$$

Suppose that all variables are binary, that s is Bernoulli with parameter $\frac{1}{2}$, that $\mathbb{P}(X_t = 1|X_s) = 1 + \varepsilon X_s - \varepsilon(1 - X_s)$, that X_w indicates if $X_s = X_a$, and that the objective is to maximize $\mathbb{E}_\delta(X_w)$, and has value $\frac{1}{2} + \varepsilon$. An optimal policy consists in choosing $X_a = X_t$. An optimal solution of the linear relaxation of MILP (8.6) on $\overline{\mathcal{P}}$ without McCormick inequalities, has value 1. Whereas an optimal solution with McCormick inequality and $b_{st}(x_s, x_t) = \frac{1}{2} + \varepsilon \mathbb{1}_{x_s = x_t}$ has value $\frac{1}{2} + \varepsilon$. However, on this simple example, the McCormick inequalities are implied by the valid inequalities of Section 8.2.

This is no more the case on the influence diagram of Figure 8.3.b, where r is a Bernoulli of parameter 0.5 and $X_s = X_r X_b + (1-X_r)(1-X_b)$, and the remaining of the parameters are defined as previously. Using the same bounds, this new example leads to exactly the same results as before.

8.3.3 Algorithm to compute good quality bounds

This section provides a dynamic programming equation to compute bounds $b_{\check{C}_v}$ on $\mu_{\check{C}_v}$ that are smaller than 1. Let $\mathcal{G}$ be a RJT, and $C_1, \dots, C_n$ be a topological order on $\mathcal{G}$. Let C_k be a vertex in $\mathcal{G}$, C_j be the parent of C_k and C_i the parent of C_j $(i < j < k)$. If $k = 1$, then $C_i = C_j = C_k = C_1$. We introduce the notation $C_j^a = (C_j \backslash (C_i \cup C_k)) \cap V^{\mathrm{a}}$. We define inductively on k the functions $\tilde{b}_k : \mathcal{X}_{C_k} \to [0,1]$ as follows.

$$\begin{cases} \tilde{b}_1(x_{C_1}) &= \prod\limits_{v \in C_1 \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \\ \tilde{b}_k(x_{C_k}) &= \Bigg(\sum\limits_{x_{\mathrm{pa}(C_j^a)}} \max\limits_{x_{C_j^a}} \sum\limits_{x_{(C_i \cup C_j)\backslash (C_k \cup C_j^a)}} \tilde{b}_i(x_{C_i}) \prod\limits_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \Bigg) \text{for } k > 1 \end{cases}$$

Proposition 8.11. *Let $\boldsymbol{\mu}$ be in $\mathcal{M}(G)$. We have $\mu_{C_k}(x_{C_k}) \leqslant \tilde{b}_k(x_{C_k})$ for all i and x_{C_k} in $\mathcal{X}_{C_k}$.*

As a consequence, $b_{\check{C}_v}$ defined as $\sum_{x_v} \tilde{b}_{C_v}$ provides an upper bound on $\mu_{\check{C}_v}$ that can be used in McCormick constraints.

Proof of Proposition 8.11. We prove the result by induction. Let $(\boldsymbol{\mu}, \boldsymbol{\delta})$ be a feasible solution of NLP 8.2 and C_j be the parent of C_k in $\mathcal{G}$, and C_i the parent of C_j $(i < j < k)$.

If $k = 1$, then the result is obtained by using $\delta_v \leqslant 1$ for all $v \in V^{\mathrm{a}}$.

We assume now that the induction is true until $k > 1$. We have

$$\begin{aligned} &\mu_{C_k}(x_{C_k}) \\ &= \sum_{x_{(C_i \cup C_j)\backslash C_k}} \mu_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{a}}} \delta_v(x_v | x_{\mathrm{pa}(v)}) && (8.14) \\ &\leqslant \max_{\delta_{(C_j \cup C_k)\backslash C_i}} \sum_{x_{(C_i \cup C_j)\backslash C_k}} \mu_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{a}}} \delta_v(x_v | x_{\mathrm{pa}(v)}) && (8.15) \\ &\leqslant \max_{\delta_{C_j \backslash (C_i \cup C_k)}} \sum_{x_{(C_i \cup C_j)\backslash C_k}} \mu_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \prod_{v \in (C_j \backslash (C_i \cup C_k)) \cap V^{\mathrm{a}}} \delta_v(x_v | x_{\mathrm{pa}(v)}) && (8.16) \\ &\leqslant \max_{\delta_{C_j \backslash (C_i \cup C_k)}} \sum_{x_{(C_i \cup C_j)\backslash C_k}} b_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}) \prod_{v \in (C_j \backslash (C_i \cup C_k)) \cap V^{\mathrm{a}}} \delta_v(x_v | x_{\mathrm{pa}(v)}) && (8.17) \end{aligned}$$

From (8.14) to (8.15), we maximize over the policies in $(C_i \cup C_j)\backslash C_k$. From (8.15) to (8.16), we bound all policies in $C_k \cap V^{\mathrm{a}}$ by 1. Then (8.17) is obtained by using the induction assumption. Let $\alpha : \mathcal{X}_{\mathrm{fa}(C_j^a)} \to \mathbb{R}$ be such that for all $x_{C_j^a} \in \mathcal{X}_{C_j^a}$,

$$\alpha(x_{C_j^a}) = \sum_{x_{(C_i \cup C_j)\backslash (C_k \cup \mathrm{fa}(C_j^a))}} \tilde{b}_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k)\backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)}).$$

Then, (8.17) becomes

$$\mu_{C_k}(x_{C_k}) \leqslant \max_{\delta_{C_j^a}} \sum_{x_{\mathrm{fa}(C_j^a)}} \alpha(x_{\mathrm{fa}(C_j^a)}) \prod_{v \in C_j^a} \delta_v(x_v | x_{\mathrm{pa}(v)})$$

Now, we can suppose that offspring$(C_v) = \{v\}$. Therefore, $|C_j^a| \leqslant 1$ and the maximum above can be decomposed into the sum.

$$\begin{aligned}
\mu_{C_k}(x_{C_k}) &\leqslant \sum_{x_{\mathrm{pa}(C_j^a)}} \max_{\delta_{C_j^a}} \sum_{x_{C_j^a}} \alpha(x_{\mathrm{fa}(C_j^a)}) \prod_{v \in C_j^a} \delta_v(x_v | x_{\mathrm{pa}(v)}) && (8.18)\\
&\leqslant \sum_{x_{\mathrm{pa}(C_j^a)}} \max_{x_{C_j^a}} \alpha(x_{\mathrm{fa}(C_j^a)}) && (8.19)
\end{aligned}$$

where from (8.18) to (8.19) we use a local maximization. Therefore, we obtain the result

$$\mu_{C_k}(x_{C_k}) \leqslant \sum_{x_{\mathrm{pa}(C_j^a)}} \max_{x_{C_j^a}} \sum_{x_{(C_i \cup C_j) \backslash (C_k \cup \mathrm{fa}(C_j^a))}} \tilde{b}_{C_i}(x_{C_i}) \prod_{v \in ((C_j \cup C_k) \backslash C_i) \cap V^{\mathrm{s}}} p(x_v | x_{\mathrm{pa}(v)})$$

□

Note that $\tilde{b}_k(x_{C_k})$ is computed via an order two recursion from $\tilde{b}_i(x_{C_i})$ where i is the grandparent of k, which can be generalized to higher order if stricter bound are needed.

8.4 Strength of the relaxations and their interpretation in terms of graph

In this section we provide interpretations of the linear relaxations of MILP (8.6) and MILP (8.9) in terms of graphs. Given an influence diagram $G = (V, A)$, we introduce the sets of arcs and influence diagrams

$$\begin{aligned}
\overline{A} &= A \cup \{(u, v) : v \in V^{\mathrm{a}} \text{ and } u \in C_v \backslash \mathrm{fa}(v)\}, & \overline{G} &= (V, \overline{A}),\\
A^{\perp\!\!\perp} &= A \cup \{(u, v) : v \in V^{\mathrm{a}} \text{ and } u \in C_v^{\not\perp\!\!\perp} \backslash \mathrm{fa}(v)\} \quad \text{and} & G^{\perp\!\!\perp} &= (V, A^{\perp\!\!\perp}).
\end{aligned}$$

Figure 8.4 illustrates $\overline{G}$ and $G^{\perp\!\!\perp}$ on the influence diagram of Figure 6.2b. Note that $A \subseteq A^{\perp\!\!\perp} \subseteq \overline{A}$, and remark the three following facts on $\overline{G}$ and $G^{\perp\!\!\perp}$. First, the definition of both influence diagrams depends on G and $\mathcal{G}$. Second, $\mathcal{G}$ is still an RJT on $\overline{G}$ and $G^{\perp\!\!\perp}$. And third, any parametrization $(\mathcal{X}_V, \mathfrak{p}, \mathbf{r})$ of G is also a parametrization of $\overline{G}$ and of $G^{\perp\!\!\perp}$. The second and third results are satisfied by any influence diagram $G' = (V, A \cup A')$, where A' contains only arcs of the form (u, v) with $v \in V^{\mathrm{a}}$ and $u \in C_v$. Denoted by $\Delta_{G'}$ the set of feasible strategies for $(G', \mathcal{X}_V, \mathfrak{p}, \mathbf{r})$, we can extend the definition of $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ in Corollary 8.2 to such G' with

$$\mathcal{M}_{\mathcal{G}}(G', \boldsymbol{\rho}) = \{\boldsymbol{\mu} \in \overline{\mathcal{P}} : \exists \boldsymbol{\delta} \in \Delta_{G'}, \mu_{C_v} = \mu_{\check{C}_v} \delta_{v | \mathrm{pa}_{G'}(v)} \text{ for all } v \text{ in } V^{\mathrm{a}}\}.$$

Using this definition, the following theorem gives an interpretation of the linear relaxations of (8.6) and (8.9).

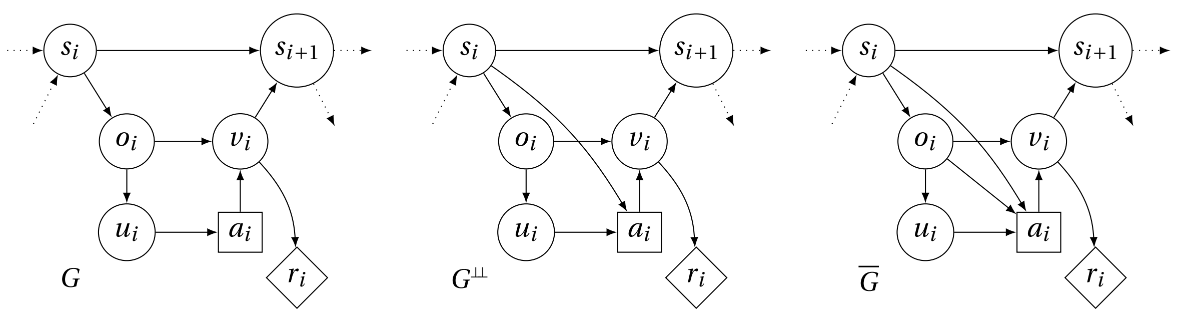


Figure 8.4 – Graph relaxations corresponding to linear relaxations for the chess game example.

Theorem 8.12. *We have*

$$\overline{\mathcal{P}} = \mathcal{M}(\overline{G}) \quad \textit{and} \quad \max_{\boldsymbol{\mu}\in\overline{\mathcal{P}}} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle = \mathrm{MEU}(\overline{G}, \boldsymbol{\rho}),$$

and

$$\mathcal{P}^{\perp\!\!\!\perp} = \mathcal{M}(G^{\perp\!\!\!\perp}) \quad \textit{and} \quad \max_{\boldsymbol{\mu}\in\mathcal{P}^{\perp\!\!\!\perp}} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle = \mathrm{MEU}(G^{\perp\!\!\!\perp}, \boldsymbol{\rho}).$$

Hence, if $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a solution of the linear relaxation of (8.6), then $\boldsymbol{\delta}$ is a strategy on $\overline{G}$, while if $(\boldsymbol{\mu},\boldsymbol{\delta})$ is a solution of the linear relaxation of (8.9), then $\boldsymbol{\delta}$ is a strategy on $G^{\perp\!\!\!\perp}$. To prove Theorem 8.12, we will need the following lemma.

Lemma 8.13. *Let v be a vertex in V^{a}. Then $x_{C_v} \mapsto p_{C_v^{\perp\!\!\!\perp}|C_v^{\not\perp\!\!\!\perp}}(x_{C_v^{\perp\!\!\!\perp}}|x_{C_v^{\not\perp\!\!\!\perp}\backslash v}, x_v)$ is a function of $(x_{C_v^{\perp\!\!\!\perp}}, x_{C_v^{\not\perp\!\!\!\perp}\backslash v})$ only. Hence, if a distribution μ_{C_v} satisfies $\mu_{C_v} = \mu_{C_v^{\not\perp\!\!\!\perp}} p_{C_v^{\perp\!\!\!\perp}|C_v^{\not\perp\!\!\!\perp}}$, then $C_v^{\perp\!\!\!\perp} \perp v \mid C_v^{\not\perp\!\!\!\perp}\backslash\{v\}$ according to μ_{C_v}.*

Proof. Consider the augmented model $\mathbb{P}_{G^\dagger}$. Let P be a $C_v^{\perp\!\!\!\perp}$-v trail. Let Q be the trail P followed by the arc (v,ϑ_v). Given that v has no descendants in C_v (because we consider the case of a gradual RJT so that offspring$(C_v) = \{v\}$), the vertex v is a v-structure of Q. As $v \in C_v^{\not\perp\!\!\!\perp}$, if P is active given $C_v^{\not\perp\!\!\!\perp}\backslash\{v\}$, then P is active given $C_v^{\not\perp\!\!\!\perp}$, which contradicts the definition of $C_v^{\perp\!\!\!\perp}$. Hence, $C_v^{\perp\!\!\!\perp} \perp v \mid C_v^{\not\perp\!\!\!\perp}\backslash\{v\}$ according to $\mathbb{P}_{G^\dagger}$, and $x_{C_v} \mapsto p_{C_v^{\perp\!\!\!\perp}|C_v^{\not\perp\!\!\!\perp}}(x_{C_v^{\perp\!\!\!\perp}}|x_{C_v^{\not\perp\!\!\!\perp}\backslash v}, x_v)$ is a function of $(x_{C_v^{\perp\!\!\!\perp}}, x_{C_v^{\not\perp\!\!\!\perp}\backslash v})$ only. The second part of the lemma is an immediate corollary. □

We can now prove Theorem 8.12.

Proof of Theorem 8.12. First, remark that, once we have proved $\overline{\mathcal{P}} = \mathcal{M}(\overline{G})$ and $\mathcal{P}^{\perp\!\!\!\perp} = \mathcal{M}(G^{\perp\!\!\!\perp})$, the result follows from Theorem 6.1.

We now prove $\overline{\mathcal{P}} = \mathcal{M}(\overline{G})$. Let μ be in $\overline{\mathcal{P}}$. Then μ is a vector of moments in the marginal polytope of the RJT $\mathcal{G}$ on $\overline{G}$. We now prove by disjunction of cases that, according to μ_{C_v}, X_v is independent from its non-descendants in $\overline{G}$ that are in C_v given $\mathrm{pa}_{\overline{G}}(v)$. If $v \in V^{\mathrm{a}}$, we have $\mathrm{fa}_{\overline{G}}(v) = C_v$ and the result is immediate. If $v \in V^{\mathrm{s}}$, we have $\mu_{C_v} = \mu_{\check{C}_v} p_{v|\mathrm{pa}_G(v)}$, for $v \in V^{\mathrm{s}}$, and $\mathrm{pa}_{\overline{G}}(v) = \mathrm{pa}_G(v)$ for v in V^{s} then gives the result. Theorem 7.3 then ensures that $\boldsymbol{\mu}$ is a vector of moments of a distribution that factorizes on $\overline{G}$, which yields $\overline{\mathcal{P}} \subseteq \mathcal{M}(\overline{G})$. The inclusion $\mathcal{M}(\overline{G}) \subseteq$

$\overline{\mathcal{P}}$ is immediate. Consider now a vector of moments $\boldsymbol{\mu}$ in $\mathcal{P}^{\perp\!\!\perp}$. Given $v \in V^{\mathrm{a}}$, Lemma 8.13 and the definition of $G^{\perp\!\!\perp}$ ensure that, according to μ_{C_v}, variable X_v is independent from its non-descendants in $G^{\perp\!\!\perp}$ in C_v, *i.e.,* $C_v^{\perp\!\!\perp}\backslash\{v\}$, given its parents in $G^{\perp\!\!\perp}$, *i.e.,* $C_v^{\not\perp\!\!\perp}\backslash v$. If $v \in V^{\mathrm{s}}$, constraints $\mu_{C_v} = \mu_{\check{C}_v} p_{v|\mathrm{pa}(v)}$ still implies that X_v is independent from its non-descendants in C_v given its parents according to μ_{C_v}, because by definition of $G^{\perp\!\!\perp}$, for $v \in V^{\mathrm{s}}$, we have $\mathrm{pa}_{G^{\perp\!\!\perp}}(v) = \mathrm{pa}_G(v)$. Theorem 7.3 again enables to conclude that $\mathcal{P}^{\perp\!\!\perp} \subseteq \mathcal{M}(G^{\perp\!\!\perp})$. Inclusion $\mathcal{M}(G^{\perp\!\!\perp}) \subseteq \mathcal{P}^{\perp\!\!\perp}$ is immediate. □

Remark furthermore that $\mathcal{M}(G')$ is generally not a polytope. Indeed, when $G' = G$, this is the reason why (8.2) is not a linear program. An important result of the theorem is that $\mathcal{M}(\overline{G})$ and $\mathcal{M}(G^{\perp\!\!\perp})$ are polytopes, and $\mathrm{MEU}(\overline{G},\boldsymbol{\rho})$ and $\mathrm{MEU}(G^{\perp\!\!\perp},\boldsymbol{\rho})$ can therefore be solved using the linear programs $\max_{\boldsymbol{\mu}\in\overline{\mathcal{P}}} \sum_{v\in V^{\mathrm{r}}} \langle r_v,\mu_v\rangle$ and $\max_{\boldsymbol{\mu}\in\mathcal{P}^{\perp\!\!\perp}} \sum_{v\in V^{\mathrm{r}}} \langle r_v,\mu_v\rangle$ respectively. In Chapter 9, we will characterize the influence diagrams for which the set of achievable moments is a polytope.

8.5 Integer programming using value functions

The aim of this section is to show the formulation that models $\mathrm{MEU}(G,\boldsymbol{\rho})$ and involving the value functions introduced in Section 6.2.3. While the approach of Section 8.1 is based on a representation of the set of achievable moments $\mathcal{M}(G)$ using linear programs, we propose in this section a representation of the set of achievable value functions $\mathcal{F}(G)$ using linear programs. We start by proving how NLP (6.8) models $\mathrm{MEU}(G,\boldsymbol{\rho})$ in Section 8.5.1, and then linearize it into MILP (6.9) in Section 8.5.2. This formulation has been introduced for the POMDP example in Section 4.4. The proofs are simpler than the ones using the vector of moments. However, the numerical experiments in Section 4.5 show that, in general, it takes longer to solve MILP (6.9) than MILP (6.6). These formulations using value functions will play a key role in Chapter 9.

8.5.1 An exact nonlinear formulation

Consider an influence diagram $G = (V^{\mathrm{s}}, V^{\mathrm{a}}, A)$. Let $\boldsymbol{\rho} = (\mathcal{X}_V, \mathfrak{p}, \mathbf{r})$ be a parametrization such that $\mathcal{X}_V = \prod_{v\in V}\mathcal{X}_v$ the support of the vector of random variables attached to all vertices of G, $\mathfrak{p} = \{p_{v|\mathrm{pa}(v)}\}_{v\in V^{\mathrm{s}}}$ is the collection of fixed and assumed known conditional probabilities, and $\mathbf{r} = \{r_v\}_{v\in V^{\mathrm{r}}}$ is the collection of reward functions $r_v\colon \mathcal{X}_v \to \mathbb{R}$. We recall that unless $\mathcal{G}$ has two disjoint connected components, without loss of generality we can assume that $\mathcal{G}$ has a single root vertex C_{v_0} with $v_0 \in V^{\mathrm{s}}$.

We recall NLP (6.8)

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \quad & \sum_{x_0\in\mathcal{X}_0} p_0(x_0)\lambda_{C_0}(x_0)\\
\text{s.t.} \quad & \lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in V^{\mathrm{s}}\cap \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in V^{\mathrm{s}}\cap \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u}\delta_{u|\mathrm{pa}(u)}, \quad \forall v\in V.
\end{aligned}$$

We have Theorem 6.5 which we recall here with the notation introduced above:

Theorem 6.5. *Let* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *be a feasible solution of NLP* (6.8)*. Then* $\boldsymbol{\lambda}$ *is the vector of value functions of the probability distribution* $\mathbb{P}_{\boldsymbol{\delta}}$ *induced by* $\boldsymbol{\delta}$*. Furthermore,* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *is an optimal solu-*

tion of NLP (6.8) *if and only if* $\boldsymbol{\delta}$ *is an optimal policy of* $\mathrm{MEU}(G,\boldsymbol{\rho})$. *In particular, NLP* (6.8) *and* $\mathrm{MEU}(G,\boldsymbol{\rho})$ *have the same optimal value.*

Theorem 6.5 gives the following corollary:

Corollary 8.14. *Given any gradual RJT* $\mathcal{G}$ *and any parametrization* $\boldsymbol{\rho}$ *on G, the set of achievable value functions* $\mathcal{F}_{\mathcal{G}}(G,\boldsymbol{\rho})$ *can be written:*

$$\mathcal{F}_{\mathcal{G}}(G,\boldsymbol{\rho}) = \left\{ \boldsymbol{\lambda}\text{s.t. } \exists \boldsymbol{\delta} \in \Delta : \left| \begin{aligned} &\lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} \\ &+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} \delta_{u|\mathrm{pa}(u)},\ \forall x_{C_v} \in \mathcal{X}_{C_v},\ \forall v \in V \end{aligned} \right. \right\}$$

Corollary 8.14 enables to write NLP (6.8) as

$$\max_{\boldsymbol{\lambda}\in\mathcal{F}(G)} \sum_{v\in V^{\mathrm{r}}} \langle r_0, \lambda_0 \rangle,$$

and it will be useful in the proofs Chapter 9 because it characterizes the set of achievable value functions by the set of constraints of (6.8).

Proof of Corollary 8.14. Let $\boldsymbol{\lambda}$ be a vector such that there exists $\boldsymbol{\delta} \in \Delta$ and

$$\lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} \delta_{u|\mathrm{pa}(u)}.$$

Hence, $(\boldsymbol{\lambda},\boldsymbol{\delta})$ is a feasible solution of NLP (6.8). Theorem 6.5 ensures that $\boldsymbol{\lambda}$ is the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$. Therefore, $\boldsymbol{\lambda} \in \mathcal{F}(G)$.

Let $\boldsymbol{\lambda} \in \mathcal{F}(G)$. Hence, there exists $\boldsymbol{\delta} \in \Delta$ such that for any $v \in V$ and $x_{C_v} \in \mathcal{X}_{C_v}$ we have:

$$\begin{aligned}
\lambda_{C_v}(x_{C_v}) &= \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in\overline{\mathrm{des}}(C_v)} r_C(X_C) | X_{C_v} = x_{C_v} \Big] \\
&= r_{C_v}(x_{C_v}) + \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in\mathrm{des}(C_v)} r_C(X_C) | X_{C_v} = x_{C_v} \Big] \\
&= r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V:\\ C_u\in \mathrm{ch}(C_v)}} \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in\overline{\mathrm{des}}(C_u)} r_C(X_C) | X_{C_v} = x_{C_v} \Big] \\
&= r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u\in\mathcal{X}_u} \mathbb{P}_{\boldsymbol{\delta}}(X_u = x_u | X_{C_v} = x_{C_v}) \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in\overline{\mathrm{des}}(C_u)} r_C(X_C) | X_{C_v} = x_{C_v}, X_u = x_u \Big] \\
&= r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u\in\mathcal{X}_u} \mathbb{P}_{\boldsymbol{\delta}}(X_u = x_u | X_{\mathrm{pa}(u)} = x_{\mathrm{pa}(u)}) \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in\overline{\mathrm{des}}(C_u)} r_C(X_C) | X_{C_u} = x_{C_u} \Big]
\end{aligned}$$

where the last equality comes from the fact that $\big(X_u \perp\!\!\!\perp X_{V\backslash \mathrm{des}(u)} | X_{\mathrm{pa}(u)}\big)_{\mathbb{P}_{\boldsymbol{\delta}}}$ and $\big(X_{\overline{\mathrm{des}}(C_u)} \perp\!\!\!\perp X_{C_v} | X_{C_u}\big)_{\mathbb{P}_{\boldsymbol{\delta}}}$. Since $\mathbb{P}_{\boldsymbol{\delta}}\big(X_v | X_{\mathrm{pa}(v)}\big) = p_{v|\mathrm{pa}(v)}$ if $v \in V^{\mathrm{s}}$ and $\mathbb{P}_{\boldsymbol{\delta}}\big(X_v | X_{\mathrm{pa}(v)}\big) = \delta_{v|\mathrm{pa}(v)}$ if $v \in V^{\mathrm{a}}$, we obtain that $(\boldsymbol{\lambda},\boldsymbol{\delta})$ satisfies

$$\lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} \delta_{u|\mathrm{pa}(u)},$$

which achieves the proof. □

Proof of Theorem 6.5. Let now $(\boldsymbol{\lambda},\boldsymbol{\delta})$ be a feasible solution of 6.8. Then $\boldsymbol{\delta}$ is a feasible solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$. We now prove that $\boldsymbol{\lambda}$ corresponds to the vector of value functions of the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$, from which we can deduce that $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\big] = \sum_{x_{v_0}} p(x_{v_0})\lambda_{C_0}(x_{v_0})$. We prove by induction that for each vertex $C\in\mathcal{V}$, $\lambda_C(x_C) = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C)}\sum_{x_{C'\backslash C}} r_{C'}(X_{C'})|X_C = x_C\right]$. Let $\succ_{\mathcal{G}}$ be a topological ordering on the RJT $\mathcal{G}$. We denote the set of vertices by $\mathcal{V} = \{C_1,\ldots,C_n\}$ where $C_i \preccurlyeq C_j$ if, and only if, $i\leqslant j$. We want to prove the induction hypothesis $\lambda_{C_i}(x_{C_i}) = \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C'\in\overline{\mathrm{des}}(C_i)}\sum_{x_{C'\backslash C_i}} r_{C'}(X_{C'})|X_{C_i} = x_{C_i}\Big]$ for $i\in\{1,\ldots,n\}$. If $i=n$, then C_n is a leaf of $\mathcal{G}$, and $\mathrm{des}(C_n)=\emptyset$. Hence, constraints (6.8b) ensure that $\lambda_{C_n}(x_{C_n}) = r_{C_n}(x_{C_n})$. Now we assume the induction hypothesis is true until $i+1$ for $1<i<n$. By definition of a topological ordering if $C_k\in\mathrm{des}(C_i)$, then $k\geqslant i$. Constraints (6.8b) ensure that

$$\lambda_{C_i} = r_{C_i}(x_{C_i}) + \sum_{\substack{k\geqslant i:\\ C_k\in\mathrm{ch}(C_i)}}\sum_{x_k} f_k(x_k|x_{\mathrm{pa}(k)})\lambda_{C_k}(x_{C_k}),$$

where $f_k(x_k|x_{\mathrm{pa}(k)}) = \delta_k(x_k|x_{\mathrm{pa}(k)})$ when $k\in V^{\mathrm{a}}$ and $f_k(x_k|x_{\mathrm{pa}(k)}) = p_k(x_k|x_{\mathrm{pa}(k)})$ otherwise. The induction hypothesis ensures that $\lambda_{C_k}(x_{C_k}) = \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'})|X_{C_k} = x_{C_k}\right]$ for all $k\geqslant i$. Therefore, we obtain

$$\begin{aligned}\lambda_{C_i}(x_{C_i}) &= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}}\sum_{x_k} f_k(x_k|x_{\mathrm{pa}(k)})\mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(X_{C'})|X_{C_k} = x_{C_k}\right]\\ &= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}}\sum_{x_k} f_k(x_k|x_{\mathrm{pa}(k)})\sum_{x_{\mathrm{des}(C_k)}}\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\overline{\mathrm{des}}(C_k)} = x_{\overline{\mathrm{des}}(C_k)}|X_{C_k} = x_{C_k}\Big)\sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'}).\end{aligned}$$

We compute the conditional probabilities $\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\overline{\mathrm{des}}(C_k)} = x_{\overline{\mathrm{des}}(C_k)}|X_{C_k} = x_{C_k}\Big)$ as follows

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\overline{\mathrm{des}}(C_k)} = x_{\overline{\mathrm{des}}(C_k)}|X_{C_k} = x_{C_k}\Big) &= \frac{\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\overline{\mathrm{des}}(C_k)} = x_{\overline{\mathrm{des}}(C_k)}\Big)}{\mathbb{P}_{\boldsymbol{\delta}}\big(X_{C_k} = x_{C_k}\big)}\\ &= \frac{\prod_{j\geqslant k:C_j\in\overline{\mathrm{des}}(C_k)} f_j(x_j|x_{\mathrm{pa}(j)})\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\check{C}_k} = x_{\check{C}_k}\Big)}{f_k(x_k|x_{\mathrm{pa}(k)})\mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\check{C}_k} = x_{\check{C}_k}\Big)}\\ &= \prod_{j>k:C_j\in\mathrm{des}(C_k)} f_j(x_j|x_{\mathrm{pa}(j)})\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\lambda_{C_i}(x_{C_i}) &= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}} \sum_{x_k} f_k(x_k|x_{\mathrm{pa}(k)}) \sum_{x_{\mathrm{des}(C_k)}} \prod_{j>k: C_j\in\mathrm{des}(C_k)} f_j(x_j|x_{\mathrm{pa}(j)}) \sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'})\\
&= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}} \sum_{x_{\overline{\mathrm{des}}(C_k)\backslash C_i}} \prod_{j\geqslant k: C_j\in\overline{\mathrm{des}}(C_k)} f_j(x_j|x_{\mathrm{pa}(j)}) \sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'})\\
&= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}} \sum_{x_{\overline{\mathrm{des}}(C_k)\backslash C_i}} \mathbb{P}_{\boldsymbol{\delta}}\Big(X_{\overline{\mathrm{des}}(C_k)} = x_{\overline{\mathrm{des}}(C_k)} | X_{\check{C}_k} = x_{\check{C}_k}\Big) \sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'})\\
&= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'}) | X_{\check{C}_k} = x_{\check{C}_k}\right]\\
&= r_{C_i}(x_{C_i}) + \sum_{\substack{k>i:\\ C_k\in\mathrm{ch}(C_i)}} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C_k)} r_{C'}(x_{C'}) | X_{C_i} = x_{C_i}\right]\\
&= \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{C'\in\overline{\mathrm{des}}(C_i)} r_{C'}(x_{C'}) | X_{C_i} = x_{C_i}\right]
\end{aligned}$$

where the fifth equality comes from the fact that $X_{\overline{\mathrm{des}}(C_k)} \perp\!\!\!\perp X_{C_i} | X_{\check{C}_k}$ for any distribution $\mathbb{P}_{\boldsymbol{\delta}}$ factorizing on G. The last equality proves the result for C_i. Consequently, $\boldsymbol{\lambda}$ is the vector of value functions of $\mathbb{P}_{\boldsymbol{\delta}}$. $\square$

8.5.2 Turning the NLP into an MILP

Like NLP (8.2), NLP (6.8) is hard to solve due to the nonlinear terms $\lambda_{C_v}\delta_{v|\mathrm{pa}(v)}$ in the constraints. We recall that there always exists at least one optimal strategy which is deterministic (and therefore integral) for $\mathrm{MEU}(G,\boldsymbol{\rho})$, that is a strategy $\boldsymbol{\delta}$ satisfying (6.3). Like the approach in Section 8.1, we can therefore add integrality constraints (6.3) to (6.8). Then we replace the term $\lambda_{C_v}(x_{C_v})\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)})$ by a variable $\alpha_{C_v}(x_{C_v})$ satisfying

$$\alpha_{C_v}(x_{C_v}) = \lambda_{C_v}(x_{C_v})\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}), \quad \forall x_{C_v} \in \mathcal{X}_{C_v},\ \forall v \in V. \tag{8.22}$$

We assume that we have access to a vector of upper bound $\mathbf{b}^{\mathrm{ub}}$ and a vector of lower bound $\mathbf{b}^{\mathrm{lb}}$ of $\boldsymbol{\lambda}$ which do not depend on $\boldsymbol{\lambda}$. In Section 8.5.3, we will explain how to compute $\mathbf{b}^{\mathrm{ub}}$ and $\mathbf{b}^{\mathrm{lb}}$ Now we linearize constraints (8.22) using the following McCormick inequalities

$$\begin{aligned}
&\alpha_{C_v}(x_{C_v}) \leqslant \lambda_{C_v}(x_{C_v}), && \forall x_{C_v} \in \mathcal{X}_{C_v}, \forall v \in V\\
&\alpha_{C_v}(x_{C_v}) \geqslant \lambda_{C_v}(x_{C_v}) - b^{\mathrm{ub}}_{C_v}(x_{C_v})(1-\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)})), && \forall x_{C_v} \in \mathcal{X}_{C_v}, \forall v \in V\\
&\alpha_{C_v}(x_{C_v}) \leqslant b^{\mathrm{ub}}_{C_v}(x_{C_v})\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}), && \forall x_{C_v} \in \mathcal{X}_{C_v}, \forall v \in V\\
&\alpha_{C_v}(x_{C_v}) \geqslant b^{\mathrm{lb}}_{C_v}(x_{C_v})\delta_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)}), && \forall x_{C_v} \in \mathcal{X}_{C_v}, \forall v \in V
\end{aligned} \tag{8.23}$$

As explained in Section 8.3, these inequalities are equivalent to the bilinear constraints (8.22). Hence, the problem we obtain the following MILP

$$\begin{aligned}
\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \quad & \sum_{x_{v_0}\in\mathcal{X}_{v_0}} p_{v_0}(x_{v_0})\lambda_{C_0}(x_{v_0}) \\
\text{s.t.} \quad & \lambda_{C_v}(x_{C_v}) = r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
& + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} \alpha_{C_u}(x_{C_u}), \quad \forall x_{C_v}\in\mathcal{X}_{C_v}, \forall v\in V \\
& \left(\alpha_{C_v},\lambda_{C_v},\delta_{v|\mathrm{pa}(v)}\right) \text{ satisfies (8.23)}
\end{aligned} \tag{8.24}$$

The tightness of the linear relaxation of MILP (6.9) depends on the quality of the bounds $b^{\mathrm{ub}}_{C_v}(x_{C_v})$ and $b^{\mathrm{lb}}_{C_v}(x_{C_v})$.

8.5.3 Algorithm to compute good quality bounds

This section provides a dynamic programming equation to compute the vector of upper bounds $\mathbf{b}^{\mathrm{ub}}$ and lower bounds $\mathbf{b}^{\mathrm{ub}}$ of $\boldsymbol{\lambda}$. Let $\mathcal{G}=(\mathcal{V},\mathcal{A})$ be a gradual RJT on $G=(V,A)$. We denote by $L_{\mathcal{G}}$ be the set of leafs of $\mathcal{G}$, *i.e.*, $L_{\mathcal{G}}=\{C\in\mathcal{V},\ \mathrm{ch}(C)=\emptyset\}$. We define inductively on v the functions $b^{\mathrm{ub}}_{C_v}, b^{\mathrm{lb}}_{C_v} : \mathcal{X}_{C_v}\to\mathbb{R}$ as follows.

$$\begin{cases}
b^{\mathrm{ub}}_{C}(x_C) = b^{\mathrm{lb}}_{C}(x_C) = r_C(x_C), & \forall x_C\in\mathcal{X}_C, \forall C\in L_{\mathcal{G}} \\
b^{\mathrm{ub}}_{C_v}(x_{C_v}) = r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) & \\
+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}} \max_{x_u\in\mathcal{X}_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}), & \forall x_{C_v}\in\mathcal{X}_{C_v}, \forall v\in V \\
b^{\mathrm{lb}}_{C_v}(x_{C_v}) = r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{lb}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) & \\
+ \sum_{u\in V^{\mathrm{a}}:\ C_u\in\mathrm{ch}(C_v)} \min_{x_u\in\mathcal{X}_u} b^{\mathrm{lb}}_{C_u}(x_{C_u}), & \forall x_{C_v}\in\mathcal{X}_{C_v}, \forall v\in V
\end{cases}$$

Proposition 8.15. *If $\boldsymbol{\lambda}$ belongs to $\mathcal{F}(G)$, then $\mathbf{b}^{\mathrm{lb}}\leqslant\boldsymbol{\lambda}\leqslant\mathbf{b}^{\mathrm{ub}}$.*

Thanks to Theorem 6.5 for any feasible solution $(\boldsymbol{\lambda},\boldsymbol{\delta})$ of MILP (8.24), $\boldsymbol{\lambda}$ belongs to $\mathcal{F}^{\mathrm{d}}(G)\subseteq\mathcal{F}(G)$. Hence, we can incorporate these bounds in MILP (8.24).

Proof of Proposition 8.15. We prove the result by induction. We use the notation of the proof of Theorem 6.5. If $i=n$, then $b^{\mathrm{ub}}_{C_n}(x_{C_n}) = b^{\mathrm{ub}}_{C_n}(x_{C_n}) = r_{C_n}(x_{C_n}) = \lambda_{C_n}(x_{C_n})$ and the result is true. We assume that for $k>i$, $b^{\mathrm{lb}}_{C_k}(x_{C_k})\leqslant\lambda_{C_k}(x_{C_k})\leqslant b^{\mathrm{ub}}_{C_k}(x_{C_k})$ for all x_{C_k} in $\mathcal{X}_{C_k}$. Since $\boldsymbol{\lambda}\in\mathcal{R}(G)$, we have the following recursion equation

$$\begin{aligned}
\lambda_{C_v}(x_{C_v}) &= r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u}(x_{C_u}) \delta_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&\leqslant r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) \delta_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&\leqslant \max_{\substack{\delta_u\in\Delta_u : u\in V^{\mathrm{a}}\\ C_u\in \mathrm{ch}(C_v)}} r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \max_{\delta_{u|\mathrm{pa}(u)}\in\Delta_u} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) \delta_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&= r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \max_{x_u} \sum_{x_u} b^{\mathrm{ub}}_{C_u}(x_u, x_{\check{C}_u})
\end{aligned}$$

By replacing the max operator with min operator and by reversing the inequalities, we obtain that $\boldsymbol{\lambda} \geqslant \mathbf{b}^{\mathrm{lb}}$. □

8.5.4 Strengthening the linear relaxation

The aim of this section is to introduce an MILP, which is similar to MILP (6.9), that uses the set $C^{\perp\!\!\!\perp}$ defined in Section 8.2 and gives tighter linear relaxation in practice.

In this section, we introduce an MILP that uses the conditional probabilities $\mathbb{P}(X_{C_v^{\perp\!\!\!\perp}}|X_{C_v^{\not\perp\!\!\!\perp}})$, which appear in the valid inequalities (6.7), and leads to a linear relaxation that is tighter in practice.

Like for MILP (6.6), since $\boldsymbol{\delta}$ are continuous variables, the McCormick's inequalities (8.23) do not longer ensure that the nonlinear constraints $\alpha_{C_v} = \lambda_{C_v}\delta_{v|\mathrm{pa}(v)}$ are satisfied. Hence, the vector $\boldsymbol{\lambda}$ of a feasible solution $(\boldsymbol{\lambda},\boldsymbol{\delta})$ of the linear relaxation of MILP (6.9) is not necessary the vector of value function induced by $\mathbb{P}_{\boldsymbol{\delta}}$. Actually, like for MILP (6.6) we wish to reduce the feasible set of the linear relaxation of MILP (6.9). However, unlike MILP (6.6) we are not able to derive valid inequalities for the variables of MILP (6.9). Instead of introducing valid inequalities for MILP (6.9), we introduce a formulation that uses the conditional independences of $X_{C^{\perp\!\!\!\perp}}$ from $\theta_{V^{\mathrm{a}}}$ given $X_{C^{\not\perp\!\!\!\perp}}$, and that gives McCormick's bound on the variables $\boldsymbol{\lambda}$ that are tighter.

We introduce this formulation in three steps to obtain another formulation that uses these independences. First, we introduce another NLP that gives an optimal strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$. Second, we turn this NLP into an MILP using the McCormick's inequalities. Third, we compute a vector of lower bounds $\mathbf{b}^{\mathrm{lb,c}}$ and upper bounds $\mathbf{b}^{\mathrm{ub,c}}$ of any feasible solution and we prove that these bounds are respectively greater than $\mathbf{b}^{\mathrm{lb}}$ and smaller than $\mathbf{b}^{\mathrm{ub}}$. This approach generalizes the one we use for the POMDP with memoryless policies in Section 4.4.4.

A nonlinear formulation. We introduce the following NLP:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\delta}} \quad \sum_{x_{v_0} \in \mathcal{X}_{v_0}} p_{v_0}(x_{v_0}) \lambda_{C_0}(x_{v_0}) \tag{8.25a}$$

$$\text{s.t.} \quad \lambda_{C_v} = r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\!\!\!\perp}}, x_u} \lambda_{C_u} p_{C_u^{\perp\!\!\!\perp} | C_u^{\not\perp\!\!\!\perp}} \delta_{u|\mathrm{pa}(u)},$$

$$\forall x_{C_v} \in \mathcal{X}_{C_v}, \forall v \in V \tag{8.25b}$$

$$\boldsymbol{\delta} \in \Delta \tag{8.25c}$$

The constraints of NLP (8.25) are similar to the ones of NLP (6.8) except that Constraints (8.25b) differ from Constraints (6.8b). Indeed, we replaced $\lambda_{C_v}(x_{C_v})$ by the expected value $\sum_{x_{C_v^{\perp\!\!\!\perp}}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v^{\perp\!\!\!\perp}} = x'_{C_v^{\perp\!\!\!\perp}} | X_{C_v^{\not\perp\!\!\!\perp}} = x_{C_v^{\not\perp\!\!\!\perp}}) \lambda_{C_v}(x_{C_v^{\not\perp\!\!\!\perp}}, x'_{C_v^{\perp\!\!\!\perp}})$. It turns out a feasible solution of NLP (8.25) is not necessary a vector of value functions. Fortunately, Proposition 8.16 below ensures that despite the loss of the value function property of Theorem 6.5, any feasible strategy $\boldsymbol{\delta}$ gives the same objective value for $\mathrm{MEU}(G, \boldsymbol{\rho})$ and NLP (8.25).

Proposition 8.16. *Let* $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ *be a feasible solution of NLP* (8.25)*. Then,* $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ *satisfies*

$$\mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C' \in \overline{\mathrm{des}}(C)} r_{C'}(X_{C'})\Big] = \sum_{x_C} \mathbb{P}_{\boldsymbol{\delta}}(X_C = x_C) \lambda_C(x_C), \tag{8.26}$$

for any C *in* $\mathcal{V}$*. In particular,* $\sum_{x_0 \in \mathcal{X}_{v_0}} p(x_0) \lambda_{v_0}(x_0) = \mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{v \in V^{\mathrm{r}}} r_v(X_v)\big]$.

Proof. Let $(\boldsymbol{\lambda}, \boldsymbol{\delta})$ be a feasible solution of NLP (8.25). Using the notation of the proof of Theorem 6.5, we prove that (8.26) holds by induction from $i = n$ to $i = 1$. For $i = n$, we have $\lambda_{C_n}(x_{C_n}) = r_{C_n}(x_{C_n})$. Therefore, we obtain

$$\sum_{x_{C_n}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_n} = x_{C_n}) \lambda_{C_n}(x_{C_n}) = \sum_{x_{C_n}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_n} = x_{C_n}) r_{C_n}(x_{C_n}) = \mathbb{E}_{\boldsymbol{\delta}}\big[r_{C_n}(X_{C_n})\big],$$

which proves the case $i = n$. Now we assume that the induction hypothesis is true until $i+1$ for $1 < i < n$. By definition of a topological ordering if $C_k \in \mathrm{des}(C_i)$, then $k > i$. We compute the right-hand side of Equation (8.26) using constraints (8.25b)

$$
\begin{aligned}
\sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v})\lambda_{C_v}(x_{C_v}) &= \sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v}) r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u, x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v})\lambda_{C_u} p_{u|\mathrm{pa}(u)} \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v}) \sum_{x_{C_u^{\perp\!\!\!\perp}}, x_u} \lambda_{C_u} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}} \delta_{u|\mathrm{pa}(u)} \\
&= \sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v}) r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_{C_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x_{\check{C}_u}) p_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_{C_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x_{\check{C}_u}) \sum_{x_{C_u^{\perp\!\!\!\perp}}, x_u} \lambda_{C_u} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}} \delta_{u|\mathrm{pa}(u)} \\
&= \sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v}) r_{C_v}(x_{C_v}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C'\in \overline{\mathrm{des}}(C_u)} r_{C'}(X_{C'})\Big] \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x'_{\check{C}_u}, x_{C_u^{\perp\!\!\!\perp}}, x_u} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x_{\check{C}_u}) p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}} \delta_{u|\mathrm{pa}(u)}\lambda_{C_u}
\end{aligned}
$$

Now we compute separately the last term above

$$
\begin{aligned}
&\sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{\substack{x'_{\check{C}_u}, x_{C_u^{\perp\!\!\!\perp}},\\ x_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x_{\check{C}_u}) p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}} \delta_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&= \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{\substack{x'_{\check{C}_u}, x_{C_u^{\perp\!\!\!\perp}}\\ x_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x'_{\check{C}_u}) \frac{\mathbb{P}_{\boldsymbol{\delta}}(X_{C_u^{\perp\!\!\!\perp}} = x_{C_u^{\perp\!\!\!\perp}}, X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}})}{\mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}})} \delta_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&= \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{\substack{x'_{\check{C}_u}, x_{C_u^{\perp\!\!\!\perp}}\\ x_u}} \frac{\mathbb{P}_{\boldsymbol{\delta}}(X_{C_u^{\perp\!\!\!\perp}} = x'_{C_u^{\perp\!\!\!\perp}}, X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}})}{\mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}})} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_u^{\perp\!\!\!\perp}} = x_{C_u^{\perp\!\!\!\perp}}, X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}}) \delta_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&= \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{\substack{x'_{\check{C}_u^{\not\perp\!\!\!\perp}}, x_{C_u^{\perp\!\!\!\perp}}\\ x_u}} \underbrace{\sum_{x'_{C_u^{\perp\!\!\!\perp}}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_u^{\perp\!\!\!\perp}} = x'_{C_u^{\perp\!\!\!\perp}} | X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}})}_{=1} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_u^{\perp\!\!\!\perp}} = x_{C_u^{\perp\!\!\!\perp}}, X_{\check{C}_u^{\not\perp\!\!\!\perp}} = x'_{\check{C}_u^{\not\perp\!\!\!\perp}}) \delta_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&= \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_{C_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{\check{C}_u} = x_{\check{C}_u}) \delta_{u|\mathrm{pa}(u)}\lambda_{C_u} \\
&= \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_{C_u}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_u} = x_{C_u})\lambda_{C_u} = \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in \overline{\mathrm{des}}(C_u)} r_C(X_C)\Big]
\end{aligned}
$$

The last equality comes from the induction hypothesis. Therefore, we deduce that

$$
\sum_{x_{C_v}} \mathbb{P}_{\boldsymbol{\delta}}(X_{C_v} = x_{C_v})\lambda_{C_v}(x_{C_v}) = \mathbb{E}_{\boldsymbol{\delta}}\Big[\sum_{C\in \overline{\mathrm{des}}(C_v)} r_C(X_C)\Big]
$$

It follows that given a strategy $\boldsymbol{\delta}$, for $C = C_0$, the equality (8.26) says that the objective values of $\mathrm{MEU}(G, \boldsymbol{\rho})$ and NLP (8.25) are equal. □

Now, we are able to write a theorem for NLP (8.25) that is similar to Theorem 6.5. We denote by

$z^*_{\mathrm{vf^c}}$ the optimal value of NLP (4.24).

Theorem 8.17. *Let* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *be a feasible solution of NLP* (8.25)*. Then,* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *is an optimal solution of NLP* (8.25) *if and only if* $\boldsymbol{\delta}$ *is an optimal strategy of* $\mathrm{MEU}(G,\boldsymbol{\rho})$*. In particular, the optimal values of* $\mathrm{MEU}(G,\boldsymbol{\rho})$ *and NLP* (8.25) *are equal.*

Proof. The proof is immediate from Proposition 8.16. □

Turning NLP (8.25) **into an MILP.** Again, we can linearize the constraints (8.25b) by introducing the variables $\boldsymbol{\alpha}$ and the McCormick's inequalities (8.23). We obtain the following MILP:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\delta}} \quad \sum_{x_{v_0}\in\mathcal{X}_{v_0}} p_{v_0}(x_{v_0})\lambda_{C_0}(x_{v_0}) \tag{8.27a}$$

$$\text{s.t.} \quad \lambda_{C_v} = r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} \lambda_{C_u} p_{u|\mathrm{pa}(u)} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\!\!\!\perp}},x_u} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}\alpha_{C_u},$$

$$\forall x_{C_v}\in\mathcal{X}_{C_v},\ \forall v\in V \tag{8.27b}$$

$$\mathrm{McCormick}(\boldsymbol{\alpha},\boldsymbol{\lambda},\boldsymbol{\delta}) \tag{8.27c}$$

$$\boldsymbol{\delta}\in\Delta^{\mathrm{d}} \tag{8.27d}$$

Computing the bounds Using the McCormick's inequalities requires to compute a vector of lower bound $\mathbf{b}^{\mathrm{lb,c}}$ and upper bound $\mathbf{b}^{\mathrm{ub,c}}$ of the feasible vectors $\boldsymbol{\lambda}$ of NLP (8.25). Let $\mathcal{G}=(\mathcal{V},\mathcal{A})$ be a gradual RJT on $G=(V,A)$. We denote by $L_{\mathcal{G}}$ be the set of leafs of $\mathcal{G}$, *i.e.*, $L_{\mathcal{G}}=\{C\in\mathcal{V},\ \mathrm{ch}(C)=\emptyset\}$. We define inductively on v the functions $b^{\mathrm{lb,c}}_{C_v}, b^{\mathrm{ub,c}}_{C_v}:\mathcal{X}_{C_v}\to\mathbb{R}$ as follows.

$$\begin{cases} b^{\mathrm{lb,c}}_{C}(x_C) = b^{\mathrm{lb}}_{C}(x_C) = r_C(x_C), & \forall x_C\in\mathcal{X}_C, \forall C\in L_{\mathcal{G}} \\ b^{\mathrm{lb,c}}_{C_v}(x_{C_v}) = r_{C_v}(x_{C_v}) + \displaystyle\sum_{\substack{u:C_u\in\mathrm{ch}(C_v)\\ u\in V^{\mathrm{s}}}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) & \\ + \displaystyle\sum_{\substack{u:C_u\in\mathrm{ch}(C_v)\\ u\in V^{\mathrm{a}}}} \min_{x_u\in\mathcal{X}_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}), & \forall x_{C_v}\in\mathcal{X}_{C_v}, \forall v\in V \\ b^{\mathrm{ub,c}}_{C_v}(x_{C_v}) = r_{C_v}(x_{C_v}) + \displaystyle\sum_{\substack{u:C_u\in\mathrm{ch}(C_v)\\ u\in V^{\mathrm{s}}}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) & \\ + \displaystyle\sum_{\substack{u:C_u\in\mathrm{ch}(C_v)\\ u\in V^{\mathrm{a}}}} \max_{x_u\in\mathcal{X}_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}), & \forall x_{C_v}\in\mathcal{X}_{C_v}, \forall v\in V \end{cases}$$

Proposition 8.18. *Let* $(\boldsymbol{\lambda},\boldsymbol{\delta})$ *be a feasible solution of NLP* (8.25)*. Then,* $\boldsymbol{\lambda}$ *satisfies* $\mathbf{b}^{\mathrm{lb,c}}\leqslant\boldsymbol{\lambda}\leqslant\mathbf{b}^{\mathrm{ub,c}}$*. In addition, the bounds obtained are tighter than* $\mathbf{b}^{\mathrm{lb}}$ *and* $\mathbf{b}^{\mathrm{ub}}$*, i.e.,* $\mathbf{b}^{\mathrm{lb}}\leqslant\mathbf{b}^{\mathrm{lb,c}}$ *and* $\mathbf{b}^{\mathrm{ub}}\geqslant\mathbf{b}^{\mathrm{ub,c}}$.

As we showed in the numerical experiments in Section 4.5 for the POMDP example, the optimal value of the linear relaxation of MILP (8.27) is not greater than the optimal value of the linear relaxation of MILP (6.9).

Proof of Proposition 8.18. Let $(\boldsymbol{\lambda},\boldsymbol{\delta})$ be a feasible solution of NLP (8.25). We prove the result by induction. We use the notation of the proof of Theorem 6.5. If $i = n$, then $b^{\mathrm{lb,c}}_{C_n}(x_{C_n}) = b^{\mathrm{ub,c}}_{C_n}(x_{C_n}) = r_{C_n}(x_{C_n}) = \lambda_{C_n}(x_{C_n})$ and the result is true. We assume that for $k > i$, $b^{\mathrm{lb,c}}_{C_k}(x_{C_k}) \leqslant \lambda_{C_k}(x_{C_k}) \leqslant b^{\mathrm{ub,c}}_{C_k}(x_{C_k})$ for all x_{C_k} in $\mathcal{X}_{C_k}$. Since $\boldsymbol{\lambda}$ satisfies constraints (8.27b), we have the following recursion equation

$$\begin{aligned}
\lambda_{C_i}(x_{C_i}) &= r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} \lambda_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u, x_{C_u^{\perp\!\!\!\perp}}} \lambda_{C_u}(x_{C_u}) \delta_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) \\
&\leqslant \max_{\substack{\delta_u:\, u\in V^{\mathrm{a}}\\ C_u\in \mathrm{ch}(C_i)}} r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u, x_{C_u^{\perp\!\!\!\perp}}} \delta_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) \\
&\leqslant r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) \\
&+ \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \max_{x_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u})
\end{aligned}$$

The first inequality comes by using the induction hypothesis and the last inequality comes from the fact that

$$\max_{\delta_u} \sum_{x_u} \delta_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) = \max_{x_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}),$$

because an optimum is reached on a vertex of the simplex Δ_u. By replacing the max operator with min operator and by reversing the inequalities, we obtain that $\boldsymbol{\lambda} \geqslant \mathbf{b}^{\mathrm{lb,c}}$.

Now we prove that $\mathbf{b}^{\mathrm{ub,c}} \geqslant \mathbf{b}^{\mathrm{ub}}$. We prove the result by induction. It holds for $i = n$. We assume that the induction hypothesis holds for every $k > i$. By definition of $\mathbf{b}^{\mathrm{lb,c}}$, we have:

$$\begin{aligned}
&b^{\mathrm{ub,c}}_{C_i}(x_{C_i}) \\
&= r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \max_{x_u\in\mathcal{X}_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u}) b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) \\
&\leqslant r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \underbrace{\sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|C_u^{\not\perp\!\!\!\perp}}(x_{\check{C}_u})}_{=1} \max_{x_u\in\mathcal{X}_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) \\
&\leqslant r_{C_i}(x_{C_i}) + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_i)}} \sum_{x_u} b^{\mathrm{ub,c}}_{C_u}(x_{C_u}) p_{u|\mathrm{pa}(u)}(x_u|x_{\mathrm{pa}(u)}) + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_i)}} \max_{x_u\in\mathcal{X}_u} b^{\mathrm{ub}}_{C_u}(x_{C_u}) = b^{\mathrm{ub}}_{C_i}(x_{C_i})
\end{aligned}$$

Again, by replacing the max operator with min operator and by reversing the inequalities, we obtain that $\mathbf{b}^{\mathrm{lb,c}} \geqslant \mathbf{b}^{\mathrm{lb}}$. □

8.6 Numerical Experiments

In this thesis, we have introduced the MILP formulation (8.6) for $\mathrm{MEU}(G,\boldsymbol{\rho})$, and shown with Corollary 9.9 that, when strengthened with valid inequalities and well-chosen bounds in the McCormick constraints, the bounds provided by the linear relaxation of our formulations are better than the ones obtained by the soluble relaxations used in the literature. In this section, we study how these formulations with moment variables and value function variables behave numerically on instances of Examples 4 and 6.

Our formulations should not be seen as an alternative to SPU (introduced p. 140) since they have different objectives. SPU is a heuristic that enables to find quickly a good solution, which is generally a local optimum on our instances because these are not soluble influence diagrams. Our exact approaches are order of magnitude slower than SPU but find better solutions than SPU and prove small optimality gaps. It is therefore natural to use the two approaches sequentially and warm start the MILP solver with the solution returned by SPU, which we do in all our numerical experiments.

Given their importance to reduce the optimality gaps, we study carefully the impact of our valid inequalities on the linear relaxation bound. For notational simplicity, and since it is unambiguous, in the rest of this section, we use the same notation to refer to a given vertex of the graph and to refer to the random variable associated with this vertex.

8.6.1 Experimental settings

Experiments performed on each instance. We have introduced two elements to strengthen the linear relaxation of our MILP formulation. We remind the reader that we introduced in Equation (8.5) on page 112 the polytope $\mathcal{Q}^b$ obtained using the vector of bounds b in McCormick constraints. Also, recall from Remark 11 that, in the special case of the polytope $\mathcal{Q}^1$ obtained with $b=1$, McCormick constraints are loose. To study the impact of McCormick constraints and valid inequalities, we solve the problems $\max\{\sum_{v\in V^{\mathrm{r}}}\langle r_v,\mu_v\rangle \mid (\boldsymbol{\mu},\boldsymbol{\delta})\in\mathcal{Q},\boldsymbol{\delta}\in\Delta^{\mathrm{d}}\}$ with four different sets $\mathcal{Q}$: $\overline{\mathcal{Q}}^1=\left(\overline{\mathcal{P}}\times\Delta^d\right)\cap\mathcal{Q}^1$ (no cuts), $\overline{\mathcal{Q}}^b=\left(\overline{\mathcal{P}}\times\Delta^d\right)\cap\mathcal{Q}^b$ (McCormick only), $\mathcal{Q}^{\perp\!\!\!\perp,1}=\left(\mathcal{P}^{\perp\!\!\!\perp}\times\Delta^d\right)\cap\mathcal{Q}^1$ (independence cuts only), and $\mathcal{Q}^{\perp\!\!\!\perp,b}=\left(\mathcal{P}^{\perp\!\!\!\perp}\times\Delta^d\right)\cap\mathcal{Q}^b$ (McCormick and independence cuts). Also, we denote respectively by $\mathcal{Q}^{\mathrm{vf}}$ and $\mathcal{Q}^{\mathrm{vf},\perp\!\!\!\perp}$ the feasible set of MILP (6.9) and (8.27), where vf means “value functions.”

Instances considered. Examples 4 and 6 are multistage models. Let T denote the number of time steps of an instance. Once T has been chosen, the influence diagram G is known, and all that is left to do is to choose a parametrization $\boldsymbol{\rho}$. We consider instances such that, for all $v\in\mathrm{fa}(V^{\mathrm{a}})$, $\mathcal{X}_v$ has k_a elements, and, for all $v\in V\backslash\mathrm{fa}(V^{\mathrm{a}})$, $\mathcal{X}_s$ has k_s elements. As we explain in the next paragraph, $\mathrm{k}_s,\mathrm{k}_a$ and T control how hard the problem is. To generate a PID instance, we start by choosing $(\mathrm{k}_s,\mathrm{k}_a,T)$, which also sets the influence diagram, and then we draw uniformly on $[0,1]$ the conditional probabilities $p_{v|\mathrm{pa}(v)}$ for all $v\in V\backslash V^{\mathrm{a}}$ and $x_{\mathrm{fa}(v)}\in\mathcal{X}_{\mathrm{fa}(v)}$ and we normalize, and we draw uniformly on $[0,10]$ the rewards $r_v(x_v)$ for all $v\in V^{\mathrm{r}}$ and all $x_v\in\mathcal{X}_v$. For our results to be representative of any instance with parameters $(\mathrm{k}_s,\mathrm{k}_a,T)$, we generate 50 instances for each triplet, and report averaged results over these 50 instances.

Intrinsic difficulty of the instances considered. Solving an influence diagram requires to find an optimal strategy, which is difficult because evaluating a given strategy is already difficult in the first place, and because optimizing on the set of strategies is then also difficult. The difficulty of evaluating a strategy is the difficulty of solving an inference problem on the underlying graph. A good indicator of this difficulty is therefore the treewidth of the graph. There is no measure that characterizes the intrinsic complexity of the problem of finding an optimal strategy, but the cost of the naive approach is the number of feasible deterministic strategies [98], *i.e.,* $\left|\Delta^{\mathrm{d}}\right|$. Our instances have a moderate treewidth, 2 for Example 4 and 3 for Example 6, and are therefore not difficult from an inference point of view. But they could be a priori difficult from an optimization point of view, because $\left|\Delta^{\mathrm{d}}\right| = \prod_{v\in V^{\mathrm{a}}} \left|\mathcal{X}_v\right|^{\prod_{u\in\mathrm{pa}(v)}\left|\mathcal{X}_u\right|} = \mathrm{k}_a^{T\mathrm{k}_s}$ is large.

Size of our MILP formulations on the instances considered. The number of constraints and variables in our MILP[4] is in $O(|V|\kappa^{\omega_{\mathrm{r}}+1})$, where ω_{r} is the rooted treewidth of the influence diagram, and $\kappa = \max_{v\in V}|\mathcal{X}_v|$.. Our MILP formulations can therefore only deal with instances of moderate rooted treewidth, which can be arbitrarily larger than the treewidth. In our examples, the rooted treewidth is equal to the treewidth and no greater than 3, while $\kappa = \max(\mathrm{k}_s, \mathrm{k}_a)$, and so the size of the MILPs remains tractable for instances with large $|\Delta^{\mathrm{d}}|$.

Experimental settings. All MILPs have been written in `Julia` [18] with the `JuMP` [41] interface and solved using `Gurobi` 9.0[52] with the default settings. Experiments have been run on a server with 192Gb of RAM and 32 cores at 3.30GHz.

Reported results. The numerical results obtained on Examples 4 and 6 are reported in Table 8.1. We denote by z, z^{LR}, and z^{B} the value of the best integer solution found, the optimal value of the linear relaxation, and the best upper bound found, respectively. We define the integrality gap g_{i} as $\frac{z^{\mathrm{LR}}-z}{z^{\mathrm{LR}}}$, and the final gap g_{f} as $\frac{z^{\mathrm{B}}-z}{z^{\mathrm{B}}}$. Let z^{SPU} be the value obtained using SPU. We define the improvement with respect to SPU i_{SPU} as $i_{\mathrm{SPU}} = \frac{z-z^{\mathrm{SPU}}}{z^{\mathrm{SPU}}}$. Each line in Table 8.1 provides average values of different quantities on 20 instances with identical parameter (k_s, k_a, T). The first column specifies the value of (k_s, k_a, T) for the instances considered, the second the approximate number of admissible strategies. The third column indicates the cuts used. In the next three columns, we report the average value of g_{i}, g_{f}, i_{SPU} on the 20 instances considered. Column "Opt" provides the percentage of instances solved to optimality, and column "Time" the average computing time. All gaps are given in percent, and computing times are given in seconds. Sometimes, the time limit is reached only for some of the 20 instances, and we end up with a non-zero average final gap together with an average computing time that is smaller than the time limit.

[4] The number of constraints defining polytopes $\overline{\mathcal{Q}}^1$ and $\overline{\mathcal{Q}}^b$ is $\sum_{v\in V^{\mathrm{s}}}|\mathcal{X}_{C_v}|+3\sum_{v\in V^{\mathrm{a}}}|\mathcal{X}_{C_v}|+\sum_{(C_u,C_v)\in\mathcal{A}}|\mathcal{X}_{C_u\cap C_v}|$, where $|\mathcal{X}_{C_v}| = \prod_{u\in C_v}|\mathcal{X}_u|$ for all v in V. If we use valid cuts, then the number of constraints of polytopes $\mathcal{Q}^{\perp\perp,1}$ and $\mathcal{Q}^{\perp\perp,b}$ is $2\sum_{v\in V^{\mathrm{s}}}|\mathcal{X}_{C_v}| + 4\sum_{v\in V^{\mathrm{a}}}|\mathcal{X}_{C_v}| + \sum_{(C_u,C_v)\in\mathcal{A}}|\mathcal{X}_{C_u\cap C_v}|$.

$$\begin{tikzcd}
 & & & & & \boxed{v_1 — r_1} \\
\boxed{— s_1} \arrow[r] & \boxed{s_1 — o_1} \arrow[r] & \boxed{s_1 o_1 — u_1} \arrow[r] & \boxed{s_1 o_1 u_1 — a_1} \arrow[r] & \boxed{s_1 o_1 a_1 — v_1} \arrow[ur] \arrow[dr] & \\
 & & & & & \boxed{s_1 v_1 — s_2}
\end{tikzcd}$$

Figure 8.5 – RJT for the chess game. The element to the right of — is the offspring.

8.6.2 Bob and Alice daily chess game

We consider the chess game example represented in Figure 6.2b. The beginning of the RJT built by Algorithm 4 for this example is represented in Figure 8.5. The rooted treewidth of this problem is 3. Table 8.1 reports results on the generated instances. We can tackle large instances of this problem: We can reach optimality in less than one hour for a strategy set of size 10^{171}, and find a small provable gap on even bigger instances. Moreover, we see that the independence cuts enables to strongly reduce the gaps and the computing time, while the improved McCormick bounds yield more minor improvements. However, on this problem, our MILP formulation only marginally improves the results returned by SPU, and its main value is the bound obtained. One can observe that the results obtained using the value function formulations produce in general poorer results. In particular, the solver does not improve the best bound computed during the resolution, which is outlined by the small difference between the integrality gap and the final gap values.

8.6.3 Partially Observable Markov Decision Process with limited memory

We now consider our POMDP instances. Figure 6.1b provides the graph representation of the POMDP with limited information. The rooted treewidth of this problem is 2. This influence diagram is not soluble[5]. Figure 8.6 represents the RJT built by Algorithm 4. On this RJT, $G^{\perp\!\!\!\perp} = \overline{G}$, and thus $\mathcal{P}^{\perp\!\!\!\perp} = \overline{P}$, which is also the constraint set of the classical MDP relaxation of a POMDP, in which the decision maker knows the state s_t when he makes the decision a_t. This MDP relaxation leads to poor lower bounds. We therefore use instead the larger RJT represented in Figure 8.7. In that RJT, $C^{\perp\!\!\!\perp}_{a_t} = \{s_t\}$, so that $G^{\perp\!\!\!\perp}$ is not anymore equal to $\overline{G}$ and the valid cuts enable to enforce the independence of s_t and a_t given (s_{t-1}, a_{t-1}, o_t) for $t > 1$. Table 8.1 provides the numerical results on our instances. This example is harder to solve to optimality. SPU has worse performance as well on this example, and our formulations manage to improve the solution found by SPU. Once again the valid cuts significantly reduce the linear relaxation gap and the solving time, even on large instances. Again, one can observe that the value function formulations give poorer results in general. For large instances, we encourage to use the formulations with moment variables rather than the value function formulations.

[5]This follows from the characterization of soluble influence diagrams in Section 9.1 and the fact that $\vartheta_{a_{t-1}} \not\perp_{G^\dagger} \mathrm{des}(a_t)|\mathrm{pa}(a_t)$ for all $t \in [T]$

			Example 6: Chess game					**Example 4: POMDP**				
$(\mathrm{k}_s, \mathrm{k}_a, T)$	$\lvert\Delta^{\mathrm{d}}\rvert$	Polytope	g_{i} (%)	g_{f} (%)	i_{SPU} (%)	Opt (%)	Time (s)	g_{i} (%)	g_{f} (%)	i_{SPU} (%)	Opt (%)	Time (s)
(3,5,20)	10^{69}	Q^{vf}	4.81	4.80	0.07	5	3421	9.25	8.61	0.66	15	3061
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	1.00	0.91	0.07	20	2895	1.88	1.77	0.56	30	2558
		$\overline{Q}^1$	5.02	0.40	0.07	65	1353	8.33	4.31	0.69	25	2852
		$\overline{Q}^b$	4.60	0.42	0.07	65	1415	7.82	4.03	0.64	25	2788
		$Q^{\perp\!\!\!\perp,1}$	1.05	0.21	0.07	70	1109	2.02	1.08	0.67	60	1679
		$Q^{\perp\!\!\!\perp,b}$	0.97	0.19	0.07	70	1110	1.68	1.09	0.67	60	1568
(3,6,20)	10^{93}	Q^{vf}	4.36	4.32	0.04	0	>3600	10.64	10.06	2.22	5	3420
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	0.72	0.65	0.04	15	3094	2.21	2.15	2.11	5	3422
		$\overline{Q}^1$	4.55	0.42	0.04	25	2731	9.68	6.13	2.30	5	3420
		$\overline{Q}^b$	4.21	0.38	0.02	35	2508	9.13	5.80	2.32	5	3420
		$Q^{\perp\!\!\!\perp,1}$	0.75	0.22	0.04	50	1886	2.35	1.48	2.32	20	3005
		$Q^{\perp\!\!\!\perp,b}$	0.70	0.22	0.02	50	2122	1.93	1.45	2.32	25	2792
(3,9,20)	10^{171}	Q^{vf}	7.20	7.20	0.11	0	>3600	8.85	8.36	1.96	0	>3600
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	1.92	1.92	0.12	10	3249	3.55	3.52	1.96	5	3426
		$\overline{Q}^1$	7.52	2.29	0.12	5	3441	8.26	6.19	2.00	0	>3600
		$\overline{Q}^b$	6.91	2.41	0.11	5	3470	7.80	5.87	2.03	0	>3600
		$Q^{\perp\!\!\!\perp,1}$	1.99	1.13	0.12	15	3061	2.28	1.75	1.96	10	3258
		$Q^{\perp\!\!\!\perp,b}$	1.89	1.13	0.11	15	3061	1.93	1.64	1.95	15	3161
(3,10,20)	10^{200}	Q^{vf}	5.93	5.93	0.02	0	>3600	12.75	12.20	1.89	0	>3600
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	1.23	1.21	0.02	10	3276	4.56	4.55	1.89	5	3440
		$\overline{Q}^1$	6.21	1.83	0.02	5	3449	11.56	9.26	1.71	0	>3600
		$\overline{Q}^b$	5.78	1.73	0.02	15	3151	19.95	9.14	1.72	0	>3600
		$Q^{\perp\!\!\!\perp,1}$	1.27	0.79	0.02	15	2882	3.48	2.91	1.64	10	3243
		$Q^{\perp\!\!\!\perp,b}$	1.21	0.78	0.02	20	2882	2.97	2.88	1.68	10	3243
(4,9,20)	10^{171}	Q^{vf}	7.12	7.12	0.04	0	>3600	11.18	10.72	0.81	0	>3600
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	1.67	1.66	0.04	10	3289	3.31	3.31	0.81	0	>3600
		$\overline{Q}^1$	7.44	4.04	0.04	0	>3600	10.27	7.95	0.98	0	>3600
		$\overline{Q}^b$	6.99	4.15	0.04	0	>3600	9.57	7.78	1.04	0	>3600
		$Q^{\perp\!\!\!\perp,1}$	1.74	1.15	0.04	10	3240	2.85	2.18	0.94	10	3274
		$Q^{\perp\!\!\!\perp,b}$	1.64	1.15	0.04	10	3240	2.27	2.14	0.97	10	3263
(4,10,20)	10^{200}	Q^{vf}	7.50	7.50	0.08	0	>3600	14.78	14.33	0.55	0	>3600
		$Q^{\mathrm{vf},\perp\!\!\!\perp}$	1.76	1.76	0.08	0	>3600	4.16	4.16	0.55	0	>3600
		$\overline{Q}^1$	8.07	4.20	0.08	0	>3600	12.80	10.52	0.81	0	>3600
		$\overline{Q}^b$	7.62	4.65	0.04	0	>3600	12.05	10.41	0.86	0	>3600
		$Q^{\perp\!\!\!\perp,1}$	1.88	1.31	0.07	5	3411	4.07	3.51	0.41	0	>3600
		$Q^{\perp\!\!\!\perp,b}$	1.76	1.29	0.08	5	3410	3.44	3.38	0.51	0	>3600

Table 8.1 – Mean results on 20 randomly generated instances with a time limit of 3600s.

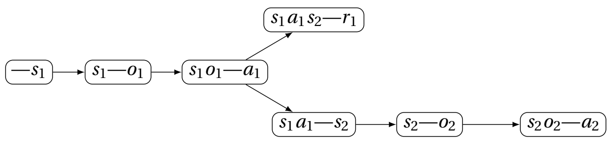


Figure 8.6 – Rooted Junction Tree built by Algorithm (4) for a POMDP with limited memory.

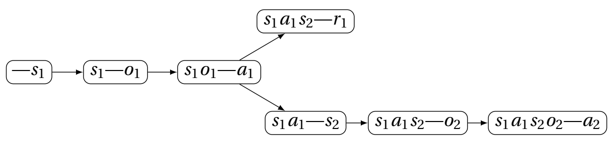


Figure 8.7 – A bigger Rooted Junction Tree for a POMDP with limited memory.

9 Polynomial cases of influence diagrams

This chapter is devoted to prove Theorems 6.7, 6.8 and 6.9. We recall the definition of a soluble influence diagram introduced in Section 6.3.

Definition 9.1. *An influence diagram G is soluble if for any parametrization $\boldsymbol{\rho}$ of G, any local optimum is a global optimum.*

Roughly speaking, a soluble influence diagram is an influence diagram that is "easy" to solve, where "easy" means that "if the inference problem is tractable, then solving the maximum expected utility problem is tractable". In the literature on influence diagrams, the soluble influence diagrams have received several characterizations. One of them is based on the notion of d-separation in directed graphical model introduced in Section 8.2.2, which enables to decide in polynomial time whether an influence diagram is soluble or not. Another characterization says that the standard SPU algorithm of Lauritzen and Nilsson [81] converges to a global optimum in a finite and polynomial number of time steps, ensuring that given an oracle solving the inference problem in polynomial time, the SPU algorithm returns an optimal solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$ in polynomial time. In this chapter, we provide a new characterization of soluble influence diagrams in terms of moments. The main result consists in saying that the convexity of the set of achievable moments is a necessary and sufficient condition of being soluble. Furthermore, we show that, for soluble influence diagrams, the linear relaxation of MILP (8.9), which is the linear relaxation of MILP (6.6) with valid cuts (6.7), gives an optimal solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$. We have even more: there are IDs which are not soluble that can be solved using the linear relaxation of MILP (8.9). In addition, we propose a linear program formulated using value functions that gives an optimal solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$ for any soluble influence diagrams G. Such a linear program turns out to be the dual of the linear relaxation of MILP (8.9).

Chapter 9 is organized as follows:

- Section 9.1 introduces several notions that are required to give the characterizations of the soluble IDs in the literature.
- Section 9.2 is devoted to prove Theorem 6.7. First, it recalls the characterizations of the soluble influence diagrams. In particular, it describes how we use the *relevance graph*, which is a notion of Koller and Friedman [75, Definition 23.9], to model the strategic dependences between the decision variables. Second, leveraging this notion, we prove Theorems 6.7 and 6.8.

- Section 9.3 gives an example showing that there are influence diagrams that are not solubles and such that the linear relaxation of MILP (8.9) gives an optimal strategy for $\mathrm{MEU}(G,\boldsymbol{\rho})$.
- Section 9.4 introduces the linear formulations (6.10), (6.11) based on the value functions variables and it proves Theorem 6.9.
- Section 9.5 presents some numerical experiments on an example of non-soluble influence diagrams that can be solved by the linear relaxation of MILP (8.9). In particular, the value obtained by running the SPU algorithm is significantly lower than the optimal value.

In this chapter, we make the assumption that influence diagrams are such that any vertex $v \in V$ has a descendant in the set of utility vertices V^{r}, i.e., $V^{\mathrm{s}} \cup V^{\mathrm{a}} = \overline{\mathrm{anc}}(V^{\mathrm{r}})$. The following remark explains why we can make this assumption without loss of generality.

Remark 13. Consider a PID $(G,\boldsymbol{\rho})$ where $G=(V,A)$ and V^{s} is the union of chance vertices V^{c} and utility vertices V^{r}. Let $(G',\boldsymbol{\rho}')$ be the influence diagram obtained by removing any vertex that is not in V^{r} and has no descendant in V^{r} and restrict $\boldsymbol{\rho}$ accordingly. If a random vector X_V factorizes as a directed graphical model on (V,A) and $V' \subseteq V$ is such that $\overline{\mathrm{anc}}(V') = V'$, then $X_{V'}$ factorizes as a directed graphical model on the subgraph induced by V' with the same conditional probabilities $p_{v|\mathrm{pa}(v)}$. Hence, given a strategy $\boldsymbol{\delta}$ on $(G,\boldsymbol{\rho})$ and its restriction $\boldsymbol{\delta}'$ to $(G',\boldsymbol{\rho}')$, we have $\mathbb{E}_{\boldsymbol{\delta}}\big[\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\big] = \mathbb{E}_{\boldsymbol{\delta}'}\big[\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\big]$ where the first expectation is taken in $(G,\boldsymbol{\rho})$ and the second in $(G',\boldsymbol{\rho}')$, and the two influence diagrams model the same $\mathrm{MEU}(G,\boldsymbol{\rho})$. △

9.1 Soluble Influence Diagrams

The aim of this section is to introduce some notions and results, whose proofs can be found in the book of Koller and Friedman [75, Chapter 23.5], and that are key in proving Theorems 6.7 and 6.8.

Consider an influence diagram $G=(V,A)$ with $V=V^{\mathrm{s}}\cup V^{\mathrm{a}}$. Three notions, *strategic relevance*, s-*reachability* and *relevance graph*, have been introduced in the literature to characterize when a local minimum is also global; see, e.g. [75, Chapter 23.5]. A decision vertex v *strategically relies* on u if the choice of a locally optimal policy δ_v given $(\delta_w)_{w\neq v}$ depends on δ_u for some parametrization $\boldsymbol{\rho}$. A decision vertex u is s-*reachable* from a decision vertex v if ϑ_u is not d-separated from $\mathrm{des}(v)$ given $\mathrm{fa}(v)$:

$$\vartheta_u \not\perp_{G^\dagger} \mathrm{des}(v) \mid \mathrm{fa}(v), \tag{9.1}$$

where $G^\dagger$ is the augmented graph defined in Section 8.2.2 and d-separation is defined in the same section. The usual definition is $\vartheta_u \not\perp_{G^\dagger} \mathrm{des}(v) \cap V^{\mathrm{r}} \mid \mathrm{fa}(v)$, but these definitions coincide in our setting, since we have assumed that $\mathrm{des}(v) \cap V^{\mathrm{r}} \neq \emptyset$ for any $v \in V^{\mathrm{a}}$. The *relevance graph* of G is the digraph H with vertex set V^{a}, and whose arcs are the pairs (v,u) of decision vertices such that u is s-reachable from v. Finally, the *single policy update* algorithm (SPU) [81] is the standard coordinate ascent heuristic for influence diagrams. It iteratively improves a strategy $\arg\max_{\delta'_v\in\Delta_v} \mathbb{E}_{\delta'_v,\delta_{-v}}\Big[\sum_{u\in V^{\mathrm{r}}} r_u(X_u)\Big]$. Koller and Friedman [75, Proposition 23.3] ensure that this local optimum can be directly derived after performing inference on the influence diagram. In particular, given an oracle solving the inference problem in polynomial time, an iteration of the

SPU algorithm runs in polynomial time.

The following proposition states that the notions of strategy relevance and s-reachability coincide in a certain sense.

Proposition 9.1. *[75, Theorems 23.2 and 23.3] Let $G = (V, A)$ be an influence diagram with $V = V^{\mathrm{s}} \cup V^{\mathrm{a}}$, and u and v be two decision vertices in V^{a}. If u is not* s*-reachable from v, then v does not strategically rely on u, while if u is* s*-reachable from v, there exists a parametrization $\boldsymbol{\rho}$ such that v strategically relies on u.*

Proposition 9.1 ensures that the relevance graph fully represents the strategic dependencies between local policies on each decision vertex. This result is key in understanding the theorem that characterizes the soluble influence diagrams, which are easily solved, and provides several equivalent criteria to identify them.

Theorem 9.2. *[75, Theorem 23.5] Given an influence diagram G, the following statements characterize a soluble influence diagram.*

1. *For any parametrization $\boldsymbol{\rho}$ of G, any local optimum is a global optimum.*
2. *For any parametrization $\boldsymbol{\rho}$ of G, SPU converges to a global optimum in a finite and polynomial number of steps.*[1]
3. *The relevance graph of G is acyclic.*

In the literature, the soluble influence diagrams are defined using the third characterization[81]. Since at each iteration of SPU algorithm, the complexity of local maximization operation only depends on the complexity of solving the inference problem, Theorem 9.2 ensures that the difficulty of solving soluble influence diagrams reduces to the difficulty of solving the inference problem on the underlying directed graphical model. In addition, the characterizations in Theorem 9.2 are key in determining if an influence diagram is soluble. In particular, it gives the following proof of Proposition 6.6.

Proof of Proposition 6.6. Thanks to Theorem 9.2, determining if an influence diagram is soluble is equivalent to determine if the relevance graph is acyclic. Building the relevance graph consists in determining for each decision vertex u, the set $\{v \in V^{\mathrm{a}} \colon \vartheta_v \not\perp \mathrm{des}(u) | \mathrm{fa}(u)\}$. The Bayes-Ball algorithm determines such a set in polynomial time $O(|V| + |A|)$ [138, Theorem 4]. Hence, we can build the relevance graph in $O\big(|V^{\mathrm{a}}|(|V| + |A|)\big)$. Checking if a directed graph is acyclic can be done in polynomial time using Kahn's algorithm [65]. Therefore, the time complexity is polynomial, which achieves the proof. □

Note that Proposition 6.6 is never mentioned or proved in the literature. However, it is common knowledge that the d-separation can be checked in polynomial time, an thus deciding if a graph is soluble can be done in polynomial time.

[1] In fact, if the graph is soluble, and if the decision vertices are ordered in reverse topological order for the relevance graph, then SPU converges after exactly one pass over the vertices.

9.2 Linear program for soluble influence diagrams

The aim of this section is to prove Theorem 6.7 (Section 9.2.1) and Theorem 6.8 (Section 9.2.2). In addition, we show that the linear relaxation of MILP (8.9) is always better than a "soluble relaxation" on the RJT, a notion we introduce in Section 9.2.3.

9.2.1 Linear relaxations

While Theorem 9.2 characterizes the soluble influence diagrams using the relevance graph, we relate this notion to the integer programs of Chapter 8 using Theorem 6.7 which we recall here:

Theorem 6.7. *If G is soluble, then there exists an RJT, such that, for every parametrization $\boldsymbol{\rho}$, an optimal solution of the linear relaxation of MILP* (6.6) *with the valid inequalities* (6.7) *induces an optimal solution of* $\mathrm{MEU}(G,\boldsymbol{\rho})$ *and both problems have the same optimal values. Such an RJT can be computed in polynomial time.*

Theorems 6.7 and 8.1 imply that, if G is soluble, then MILP (8.9) reduces to the linear program

$$\max_{\boldsymbol{\mu}\in\mathcal{P}^{\perp\!\!\perp}} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle.$$

However, it is not a sufficient condition for being soluble. Indeed, we will show in Section 9.3 that there are some influence diagrams that are not soluble and such that the linear relaxation of MILP (8.9) induces an optimal strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$.

Theorem 6.7 is a corollary of Theorem 8.12 and the following lemma.

Lemma 9.3. *There exists an RJT $\mathcal{G}$ such that $G^{\perp\!\!\perp} = G$ if and only if G is soluble. Such an RJT can be computed using Algorithm 5.*

The proof of this lemma is postponed at the end of the section because of the length of its proof. Note that based on a topological order of the relevance graph, Algorithm 5 proceeds by computing a larger graph (satisfying perfect recall) that contains the graph G and that assigns the same parent sets to elements of V^{s} as in G, then uses a topological order of this graph to order the vertices of G for the computation of a rooted junction tree.

Algorithm 5 Build a "good" RJT for a soluble graph G

```
1:  Input: An ID $G = (V, A)$.
2:  Initialize: $A' = \emptyset$.
3:  Compute the relevance graph $H$ of $G$
4:  Compute an arbitrary topological order $\preccurlyeq_H$ on $V^{\mathrm{a}}$ for the relevance graph $H$
5:  $A' \leftarrow A \cup \{(u,v) \in V^{\mathrm{a}} \times V^{\mathrm{a}} \colon u \preccurlyeq_H v\}$
6:  $G' \leftarrow (V, A')$
7:  $A'' \leftarrow A \cup \{(u,v) \in V^{\mathrm{s}} \times V^{\mathrm{a}} \colon u \notin \mathrm{des}_{G'}(v)\}$
8:  $A'' = (V, A'')$
9:  Compute an arbitrary topological order $\preccurlyeq$ on $G''$
10: Return the result of Algorithm 4 for $(G, \preccurlyeq)$
```

Proof of Theorem 6.7. Note that by definition for any feasible solution $(\boldsymbol{\mu},\boldsymbol{\delta})$ of the linear relaxation of MILP (8.9), we have $\boldsymbol{\mu} \in \mathcal{P}^{\perp\!\!\perp}$. If G is a soluble influence diagram, then Lemma 9.3 ensures that there exists an RJT $\mathcal{G}$ such that $G = G^{\perp\!\!\perp}$. Consider such an RJT $\mathcal{G}$ built by Algorithm 5. Since $G = G^{\perp\!\!\perp}$, we obtain $\mathcal{M}_{\mathcal{G}}(G) = \mathcal{M}_{\mathcal{G}}(G^{\perp\!\!\perp})$. On the other hand, Theorem 8.12 ensures that for every parametrization $\boldsymbol{\rho}$, we have $\mathcal{M}_{\mathcal{G}}(G^{\perp\!\!\perp},\boldsymbol{\rho}) = \mathcal{P}^{\perp\!\!\perp}$. We deduce that for such an RJT, we obtain $\mathcal{M}_{\mathcal{G}}(G,\boldsymbol{\rho}) = \mathcal{P}^{\perp\!\!\perp}$. Therefore, for any optimal $(\boldsymbol{\mu},\boldsymbol{\delta})$ solution of the linear relaxation of MILP (8.9), there exists a strategy $\boldsymbol{\delta}' \in \Delta$ such that $\boldsymbol{\mu}$ is the vector of moments of $\mathbb{P}_{\boldsymbol{\delta}'}$ and the optimal values are equal because

$$\max_{\boldsymbol{\mu}\in\mathcal{P}^{\perp\!\!\perp}} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle = \max_{\boldsymbol{\mu}\in\mathcal{M}_{\mathcal{G}}(G,\boldsymbol{\rho})} \sum_{v\in V^{\mathrm{r}}} \langle r_v, \mu_v\rangle = MEU(G,\boldsymbol{\rho}).$$

Algorithm 5 runs in polynomial time because each step is polynomial in the size of the graph. It achieves the proof. □

For any set C and binary relation R, we denote by $C_{\mathrm{R}v}$ the set of vertices u in C such that $u\,\mathrm{R}\,v$. The following lemma will be useful in the proof of Lemma 9.3. Let H denote the relevance graph of G.

Lemma 9.4. *In general, if $u \in \mathrm{des}(v)$, then $(v,u) \in H$. If G is soluble, then it becomes an equivalence.*

As an immediate consequence of Lemma 9.4, if G is soluble and $\preccurlyeq$ is a topological order on G, then its restriction $\preccurlyeq_H$ to V^{a} is a topological order on the relevance graph H.

Proof. Assume that u is s-reachable from v, that is (v,u) is an arc in H. We first show that this implies that u and v have descendants in common. Indeed, by definition of s-reachability, this means that there exist $w \in \mathrm{des}(v)$ and an active trail T from ϑ_u to w. Either, T is a directed path and w is also a descendant of u or T must have a v-structure. In the latter case, let x be the vertex with the v-structure closest to ϑ_u on T; since the trail is active, we must have $x \in \mathrm{fa}(v)$ but since x is a descendant of u, in that case, v must be a descendant of u. In both cases considered u and v have descendants in common. Now, if u is not a descendant of v, then v is s-reachable from u, which is not possible as H is acyclic. Hence $u \in \mathrm{des}(v)$. □

Proof of Lemma 9.3. Let G be a soluble influence diagram. We start by proving that Algorithm 5 with G as an input returns an RJT $\mathcal{G}$. It suffices to show that it is possible to compute topological orderings in step 9, that is, to prove that H, G' and G'', defined in Algorithm 5, are acyclic. H is acyclic because the influence diagram is soluble. We now prove that G' is acyclic. As G is acyclic and by definition of G', a cycle in G' contains necessarily two vertices of V^{a}. Let u and v thus be two distinct elements of V^{a}. Remark that, if there exists a path from u to v in G, then v is strategically reachable from u, and $u \preccurlyeq_H v$. Hence, by definition of G', the indexes of vertices in V^{a} for $\preccurlyeq_H$ can only increase along a path in G'. There is therefore no cycle in G' containing two vertices in V^{a}, and thus no cycle in G'. We now prove that G'' is acyclic. Suppose that there is a cycle in G''. Let $\preccurlyeq_{G'}$ be a topological order on G', and let v_h be the smallest vertex v for $\preccurlyeq_{G'}$ in the cycle such that there is an arc (u,v) in $E''\backslash E'$ in the cycle. And let (u_h,v_h) be the corresponding arc in the cycle. Let (u_l,v_l) be the arc of A' right before (u_h,v_h) in the cycle such that $v_l \in V^{\mathrm{a}}$. Arc (u_l, v_l) is possibly identical to (u_h, v_h). By definition of G', given two disjoint vertices u and v in V^{a}, either $(u,v) \in A'$ or $(v,u) \in A'$. Since $v_h \preccurlyeq_{G'} v_l$ by definition of v_h, we have either $v_h = v_l$ or $(v_h, v_l) \in A'$. And since all the arcs in the v_l-u_h subpath of the cycle are in A', we have $u_h \in \overline{\mathrm{des}}_{G'}(v_l)$. Hence $u_h \in \mathrm{des}_{G'}(v_h)$, which contradicts the definition of E'' in Step 7. Hence, Algorithm 5 always returns an RJT, which we denote by $\mathcal{G}$.

It remains to prove that $\mathcal{G}$ is such that $C_v^{\not\downarrow} \subseteq \mathrm{fa}(v)$ for each decision vertex $v \in V^{\mathrm{a}}$. We start with two preliminary results. Remark that $A \subseteq A'$ implies that $\preccurlyeq$ is a topological order on G. Let $\preccurlyeq_H$ denotes its restriction to V^{a}. Lemma 9.4 ensures that $\preccurlyeq_H$ is a topological order on H. Hence, we have

$$\vartheta_{V^{\mathrm{a}}_{\prec v}} \perp \mathrm{des}(v) \mid \mathrm{fa}(v), \quad \text{for all } v \in V^{\mathrm{a}}. \tag{9.2}$$

Furthermore, if $u \in V^{\mathrm{a}}$ and $v \in V^{\mathrm{s}}_{\not\preccurlyeq u}$, the definition of G' implies the existence of a path from u to v in G, and hence the existence of a path from $V^{\mathrm{a}}_{\not\preccurlyeq u}$ to v in G.

We now prove $C_v^{\not\downarrow} \subseteq \mathrm{fa}(v)$ for each $v \in V^{\mathrm{a}}$. This part of the proof is illustrated on Figure 9.1.a. Let v be a vertex in V^{a}, let $u \in C_v \backslash \mathrm{fa}(v)$, and let $b \in V^{\mathrm{a}}_{\prec u}$. We only have to prove that u is d-separated from ϑ_b given $\mathrm{fa}(v)$. We start by proving that u and v have common descendants. Proposition 7.9 guarantees that (7.7b) is an equivalence. Hence, there exists a u-v trail in $V_{\succ v}$. Consider such a u-v trail Q with a minimum number of $\veebar$structures. Suppose for a contradiction that Q has more than one $\veebar$structure. Starting from v, let w_1 be the first $\veebar$structure of Q and u_1 bet its first vertex with diverging arcs u_1. Using the result at the end of the previous paragraph, we have $u_1 \in \overline{\mathrm{des}}(V^{\mathrm{a}}_{\not\preccurlyeq v})$. Since Q has been chosen with a minimal number of v-structures, we obtain $u_1 \in \overline{\mathrm{des}}(V^{\mathrm{a}}_{\succ v})$. Let a_1 denote an ancestor of u_1 in $V^{\mathrm{a}}_{\succ v}$. Since $w_1 \in \mathrm{des}(v)$, Equation (9.2) ensures that $w_1 \perp \vartheta_v \mid \mathrm{fa}(a_1)$. Hence, the v-w_i path is not active given $\mathrm{fa}(a_1)$, and it therefore necessarily intersects $\mathrm{pa}(a_i)$. Hence, $u_1 \in \mathrm{des}(v)$, and Q there exists a u-v trail Q with fewer $\veebar$structures than Q, which gives a contradiction. Trail Q therefore has a unique v-structure, and u and v have a common descendant w. Hence, if ϑ_b-u trail P is active given $\mathrm{fa}(v)$, then P followed by a u-w path is active given $\mathrm{fa}(v)$. The fact that $\mathrm{des}(v) \perp \vartheta_b | \mathrm{fa}(v)$ ensures that there is no-such path P, and we have proved that u is d-separated from ϑ_b given $\mathrm{fa}(v)$.

Conversely, let G be a non-soluble influence diagram, and $\mathcal{G}$ an RJT on G. Let u and v be two vertices in V^{a} such that $\mathrm{des}(v) \not\perp \vartheta_u | \mathrm{pa}(v)$ and $\mathrm{des}(u) \not\perp \vartheta_v | \mathrm{pa}(u)$. Without loss of generality, we assume that if there is a path between C_u and C_v, it is from C_v to C_u. To prove the converse, we prove that $C_u^{\not\downarrow} \neq \mathrm{fa}(u)$. This part of the proof is illustrated on Figure 9.1.b. There exists an active trail Q from $w \in \mathrm{des}(u)$ to ϑ_v given $\mathrm{pa}(u)$. Starting from w, let x be the first vertex with diverging arcs of Q if Q contains such a structure, and be equal to v otherwise. And let P be the w-x subtrail of Q. Remark that P must be an x-w path in G, because any passing $\veebar$structure on P cannot be at a descendant of w, for it would then be a descendant of u which could not have any descendant in $\mathrm{fa}(u)$ as G is acyclic. The path P contains no $\veebar$structure, and is active given $\mathrm{fa}(u)$. Hence, it does not intersect $\mathrm{fa}(u)$. Since x and u have w as common descendant, Proposition 7.4 ensures that C_x and C_u are on the same branch of $\mathcal{G}$. If $v = x$, $x \in \mathrm{anc}(w)$ and there is a path in $\mathcal{G}$ from C_x to C_w, moreover, since we assumed C_u is a descendant of C_v in $\mathcal{G}$, and since $u \in \mathrm{anc}(w)$, then the path from C_x to C_w contains C_u and all the vertices of P. Now, if $x \neq v$, then x is the first vertex with diverging arcs, and in that case it belongs to $\mathrm{anc}(u)$, because $Q \setminus P$ must contain at least one $\veebar$structure and any such $\veebar$structure can only be at a vertex in $\mathrm{anc}(u)$. So, again, there is a path in $\mathcal{G}$ from C_x to C_w which contains C_u and all the

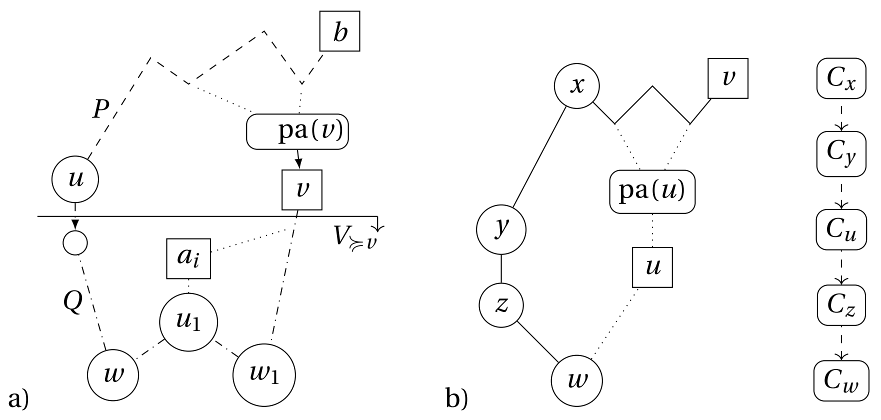


Figure 9.1 – Illustration of the proof of Lemma 9.3. a) Direct statement, with $j = i - 1$. Trail P is illustrated by dashed line, trail Q by a dash-dotted line, and other paths by dotted lines. b) Converse statement, with path Q in plain line, and other paths dotted lines, and paths in $\mathcal{G}$ in dashed lines.

vertices of P. Starting from x, let y be the last vertex of P such that C_y is above C_u in $\mathcal{G}$, and z be the child of y in P. But since Q is active, the y-ϑ_v subtrail of Q is active given $\mathrm{fa}(u)$, and we therefore have $C_u^{\not\perp\!\!\!\perp} \neq \mathrm{fa}(u)$.

□

9.2.2 Characterization using the set of achievable moments

In this section, we give another characterization of soluble influence diagrams using Theorem 6.8 which we recall here:

Theorem 6.8. *An influence diagram G is soluble if and only if there exists an RJT $\mathcal{G}$ such that for every parametrization $\boldsymbol{\rho}$ on G, the set of achievable moments $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ is a polytope. In this case, such an rooted junction tree $\mathcal{G}$ can be computed in polynomial time.*

In fact, Theorem 6.8 is a corollary of the following stronger result:

Theorem 9.5. *If G is not soluble then there exists a parametrization $\boldsymbol{\rho}$ such that, for every rooted junction tree $\mathcal{G}$, the set of achievable moments $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ is not convex.*

If G is soluble, then there exists an rooted junction tree $\mathcal{G}$ such that for every parametrization $\boldsymbol{\rho}$, the set of achievable moments $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ coincides with $\mathcal{P}^{\perp\!\!\!\perp}$. Such an rooted junction tree can be computed using Algorithm 5.

The set of achievable moments fully characterizes the soluble influence diagrams. To visualize the form of the set of achievable moments, we introduce the following the lemma:

Lemma 9.6. *For every parametrization $\boldsymbol{\rho}$ and every rooted junction tree $\mathcal{G}$, the convex hull of the set of achievable moments coincides with the convex hull of the set of achievable deterministic moments, i.e.,* $\mathrm{Conv}\big(\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})\big) = \mathrm{Conv}\Big(\mathcal{M}_{\mathcal{G}}^{\mathrm{d}}(G, \boldsymbol{\rho})\Big)$*, where the notation* $\mathrm{Conv}(A)$ *denotes the convex hull of a set A.*

Proof. Since $\mathcal{M}^{\mathrm{d}}(G) \subseteq \mathcal{M}(G)$, we have $\mathrm{Conv}\big(\mathcal{M}^{\mathrm{d}}(G)\big) \subseteq \mathrm{Conv}(\mathcal{M}(G))$.

Now it suffices to prove that $\mathcal{M}(G) \subseteq \mathrm{Conv}\big(\mathcal{M}^{\mathrm{d}}(G)\big)$. Let $\boldsymbol{\mu} \in \mathcal{M}(G)$. Corollary 8.2 ensures that there exists $\boldsymbol{\delta} \in \Delta$ such that $\mu_{C_v} = \delta_{v|\mathrm{pa}(v)}\mu_{C_v\backslash\{v\}}$ for all $v \in V^{\mathrm{a}}$. Since $\boldsymbol{\delta} \in \Delta$ and by definition

$$\Delta = \prod_{v \in V^{\mathrm{a}}} \prod_{x_{\mathrm{pa}(v)} \in \mathcal{X}_{\mathrm{pa}(v)}} \underbrace{\Big\{\delta_{v|\mathrm{pa}(v)}(.|x_{\mathrm{pa}(v)}) \in \mathbb{R}_+^{\mathcal{X}_v} : \sum_{x_v} \delta_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) = 1\Big\}}_{\Delta_v(x_{\mathrm{pa}(v)})}.$$

In addition, for any $v \in V^{\mathrm{a}}$ and $x_{\mathrm{pa}(v)} \in \mathcal{X}_{\mathrm{pa}(v)}$, we have:

$$\Delta_v(x_{\mathrm{pa}(v)}) = \mathrm{Conv}\Big(\underbrace{\Big\{\delta_{v|\mathrm{pa}(v)}(.|x_{\mathrm{pa}(v)}) \in \{0,1\}^{\mathcal{X}_v} : \sum_{x_v} \delta_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) = 1\Big\}}_{\Delta_v^{\mathrm{d}}(x_{\mathrm{pa}(v)})}\Big).$$

Therefore, we obtain that $\Delta = \prod_{v \in V^{\mathrm{a}}} \prod_{x_{\mathrm{pa}(v)} \in \mathcal{X}_{\mathrm{pa}(v)}} \mathrm{Conv}(\Delta_v^{\mathrm{d}}(x_{\mathrm{pa}(v)}))$. It is known that the convex hull of a Cartesian product is the Cartesian product of the convex hulls. Therefore,

$$\Delta = \mathrm{Conv}\Big(\prod_{v \in V^{\mathrm{a}}} \prod_{x_{\mathrm{pa}(v)} \in \mathcal{X}_{\mathrm{pa}(v)}} \Delta_v^{\mathrm{d}}(x_{\mathrm{pa}(v)})\Big).$$

On the other hand, we have $\prod_{v \in V^{\mathrm{a}}} \prod_{x_{\mathrm{pa}(v)} \in \mathcal{X}_{\mathrm{pa}(v)}} \Delta_v^{\mathrm{d}}(x_{\mathrm{pa}(v)}) = \Delta^{\mathrm{d}}$. We deduce that $\Delta = \mathrm{Conv}(\Delta^{\mathrm{d}})$.

Hence, there exists a finite set I of deterministic strategies $\boldsymbol{\delta}^i \in \Delta^{\mathrm{d}}$ for any $i \in I$ and non-negative scalars $(\lambda^i)_{i \in I}$ such that $\sum_{i \in I} \lambda^i = 1$ and $\boldsymbol{\delta} = \sum_{i \in I} \lambda^i \boldsymbol{\delta}^i$. Therefore,

$$\mu_{C_v} = \sum_{i \in I} \lambda_i \delta^i_{v|\mathrm{pa}(v)} \mu_{C_v\backslash\{v\}}.$$

Let $\tilde{\mu_{C_v}}^i = \delta^i_{v|\mathrm{pa}(v)}\mu_{C_v\backslash\{v\}}$ for all $i \in I$. Then for all $v \in V^{\mathrm{a}}$, $\mu_{C_v} = \sum_{i \in I} \lambda_i \tilde{\mu_{C_v}}^i$. For all $v \in V^{\mathrm{s}}$, we set $\tilde{\mu_{C_v}}^i = \mu_{C_v}$. We deduce that $\boldsymbol{\mu} = \sum_{i \in I} \lambda_i \tilde{\boldsymbol{\mu}}^i$ and $\tilde{\boldsymbol{\mu}}^i \in \mathcal{M}^{\mathrm{d}}(G)$. Therefore $\boldsymbol{\mu} \in \mathrm{Conv}\big(\mathcal{M}^{\mathrm{d}}(G)\big)$, which achieves the proof. □

Since $\mathrm{Conv}\big(\mathcal{M}^{\mathrm{d}}(G)\big)$ is a polytope, Lemma 9.6 gives the following corollary:

Corollary 9.7. *The set of achievable moments is convex if and only if it is a polytope.*

Although we do not use this corollary in the remainder of this thesis, it enables to provide an abstract representation of the set of achievable moments. Figure 9.2 illustrates Lemma 9.6 and gives a planar representation of the set of achievable moments.

Proof of Theorem 6.8 and Theorem 9.5. If G is soluble, Lemma 9.3 ensures that Algorithm 5 builds an RJT $\mathcal{G}$ such that $G^{\perp\!\!\perp} = G$, and Theorem 8.12 ensures that $\mathcal{P}^{\perp\!\!\perp} = \mathcal{M}(G)$.

Consider now the result for non-soluble influence diagrams. Let G be a non-soluble influence diagram. Let a and b be two decision vertices that are both strategically dependent on the other one.

First, we suppose that $a \notin \mathrm{anc}(b)$ and $b \notin \mathrm{anc}(a)$. Let P be a path from a to $w \in \mathrm{des}(b)$ with a minimum number of arcs, and Q be a b-w path with a minimum number of arcs. Then w

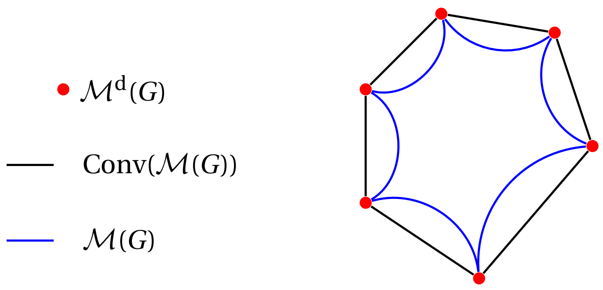
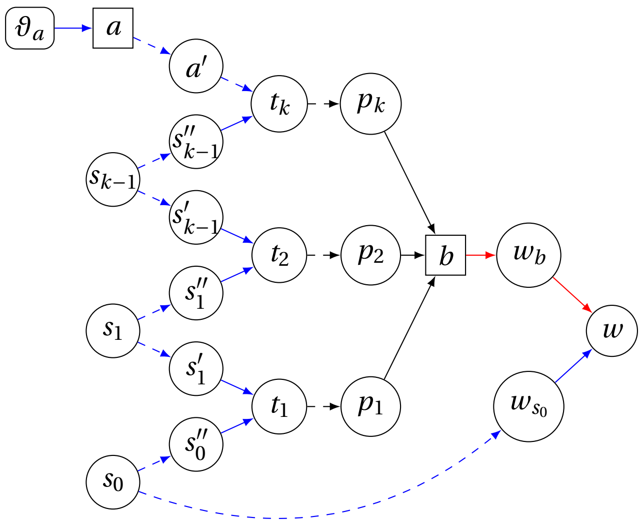

$\mathcal{M}^{\mathrm{d}}(G)$
$\mathrm{Conv}(\mathcal{M}(G))$
$\mathcal{M}(G)$

Figure 9.2 – Planar representation of the set of achievable moments $\mathcal{M}(G)$, the set of achievable deterministic moments $\mathcal{M}^{\mathrm{d}}(G)$ and the convex hull of the set of achievable moments $\mathrm{Conv}(\mathcal{M}(G)) = \mathrm{Conv}\big(\mathcal{M}^{\mathrm{d}}(G)\big)$.



$\mathcal{X}_{s_0} = \mathcal{X}_{s''_0} = \mathcal{X}_{w_{s_0}} = \{1, 2, \mathrm{e}\}$
$\mathcal{X}_{s_\ell} = \mathcal{X}_{s'_\ell} = \mathcal{X}_{s''_\ell} = \{1, 2\}, \forall \ell > 0$
$\mathcal{X}_{t_\ell} = \mathcal{X}_{p_\ell} = \{0, 1\}, \forall \ell > 0$
$\mathcal{X}_b = \mathcal{X}_{w_b} = \{1, 2, \mathrm{j}\}$
$\mathcal{X}_w = \{-10, 0, 1, 2\}$
$\xrightarrow{\quad}$ P $\xrightarrow{\quad}$ Q
$p_{s_0}(x) = 1/3$ for x in $\{1, 2, \mathrm{e}\}$
$p_{s_\ell}(x) = 1/2$ for $\ell > 0$ and $x \in \{1, 2\}$
$X_{t_\ell} = \mathbb{1}(X_{s''_{k-1}} \neq X_{s'_\ell}), \forall \ell$
$X_{s''_\ell} = X_{s_\ell}, \quad X_{s'_\ell} = X_{s_i}, \forall \ell$
$X_{p_\ell} = X_{t_\ell}, \quad X_{w_b} = X_b, \forall \ell$
$X_{w_{s_0}} = X_{s_0}$

$$X_w = \begin{cases} 0 & \text{if } X_{w_b} = \mathrm{j} \\ i & \text{if } X_{w_b} = X_{w_{s_0}} = i \text{ for } i \in \{1, 2\} \\ -10 & \text{otherwise.} \end{cases}$$

Figure 9.3 – Influence diagram and parametrization used in the proof of Theorem 9.5

is the unique vertex in the intersection of P and Q. Let u and v be the parents of w in P and Q respectively. Consider a parametrization where all the variables that are not in P or Q are unary, all the variables in P and Q are binary, all the variables in the a-u subpath of P are equal to X_a, all the variables in the b-v subpath of P are equal to X_b, and $p_{w|\mathrm{pa}(w)}$ is defined arbitrarily. Let $\mathcal{G}$ be an arbitrary junction tree, C be its cluster containing $\mathrm{fa}(w)$. Then choosing a distribution μ_a as policy δ_a and a distribution μ_b as policy δ_b implies that the restriction of μ_C to X_{uv} is $\mu_{uv} = \mu_a \mu_b$. Hence, the marginalization on X_{uv} of the set of distributions μ_C that can be reached for different policy is the set of independent distributions, which is not convex. Hence, $\mathcal{M}(G)$ is not convex.

We now consider the case where $a \in \mathrm{anc}(b)$ or $b \in \mathrm{anc}(a)$. W.l.o.g., we suppose $a \in \mathrm{anc}(b)$. There exists a trail from ϑ_a to w in $\mathrm{des}(b)$ that is active given $\mathrm{pa}(b)$. Let Q be such a trail with a minimum number of v-structures. And let P be a b-w path. W.l.o.g., we suppose that w is the only vertex in both P and Q. Let w_b be the parent of w on P and w_{s_0} its parent in Q. Starting from w, let $s_0,\ldots,s_{k-1}$ denote the vertices with divergent arcs in Q, let $t_1,\ldots,t_k$ the v-structures, and p_ℓ denote the parent of b that is below t_ℓ. Finally let s'_ℓ (resp. s''_ℓ) denote the parent of t_ℓ (resp $t_{\ell+1}$) on the s_ℓ-t_ℓ subpath (resp. s_ℓ-t_{i+1} subpath) of Q. The structures that we have just exhibited entail that G contains a subgraph of the form represented on Figure 9.3. Each dashed arrows correspond to a path whose length may be equal to 0, in which case the vertices connected by the path are the same.

We now introduce a game that we will be able to encode on the graph of Figure 9.3 and hence on G. This game is a dice game with two players a and b. Before rolling a uniform die with three faces, player a chooses to play 1 or 2, where “playing i" means observing if the die is equal to i, and passing this information to player b. The die s_0 is rolled. If a has played 1 (resp. 2), he passes the information true to b if the die s_0 is equal to 1 (resp. 2), and false if it is equal to 2 (resp. 1), or something else e. Player b does not know what a has played. Based on the information he receives, player b decides to play 1, 2, or joker, that we denote j. If he plays j, then none of the player either earns or loses money. If he plays i in $\{1,2\}$, then both players earn i euros if die s_0 is equal to i, and lose 10 euros otherwise. The goal is maximize the expected payoff. This game has two locally optimal strategies $\boldsymbol{\delta}^1$ and $\boldsymbol{\delta}^2$. In strategy $\boldsymbol{\delta}^i$, player a plays i and b plays i if he receives true and j otherwise. Both strategies are locally optimal: each players decision is the best possible given the other ones. But only strategy $\boldsymbol{\delta}^2$ is globally optimal.

It changes nothing to the game if we add $k-1$ coin tosses $X_{s_1},\ldots,X_{s_{k-1}}$, and player b observes the k equality tests $X_{t_1},\ldots,X_{t_{k-1}}$, where $X_{t_\ell} = \mathbb{1}(X_{s_{\ell-1}} = X_{s_\ell})$. Indeed, player b can compute $\sum_{\ell=1}^k x_{p_\ell}$ and knows that $X_a = X_{s_0}$ if and only if this sum is even. The parameterization of the influence diagram that enables to encode this game is specified on the right part of Figure 9.3. For any x, the mapping $\mathbb{1}_x(\cdot)$ is the indicator function of x. All the variables that are not on Figure 9.3 or on the paths on Figure 9.3 are unary. All the variables along paths represented by dashed arrows are equal. Strategies $\boldsymbol{\delta}^i$ can therefore be defined as

$$\delta_a^i = \mathbb{1}_i \quad \text{and} \quad \delta_b^i(x_{p_1},\ldots,x_{p_k}) = \begin{cases} i & \text{if } \sum_{\ell=1}^k x_{p_\ell} = 0 \bmod 2, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathbb{1}_i$ is the Dirac in i. A technical case to handle is the one where $a = a' = t_k$. In that case, we define $\mathcal{X}_a = \{0,1\}$ and $\delta_a^i = \mathbb{1}_i(X''_{s_{k-1}})$.

Consider now an RJT $\mathcal{G}$ on G. Let C be a vertex of $\mathcal{G}$ that contains $\mathrm{fa}(w)$. Then C contains both w_b and w_{s_0}. Let μ_C^1 and μ_C^2 be the distributions induced by $\boldsymbol{\delta}^1$ and $\boldsymbol{\delta}^2$ on X_C, and $\mu_{bs_0}^1$ and $\mu_{bs_0}^2$ their marginalization on $X_{w_b w_{s_0}}$. Since $X_{w_b} = X_b$ and $X_{w_{s_0}} = X_{s_0}$, $\mu_{bs_0}^1$ and $\mu_{bs_0}^2$ are the distributions induced by $\boldsymbol{\delta}^1$ and $\boldsymbol{\delta}^2$ on X_{bs_0}. Let $\mu_{bs_0} = \frac{\mu_{bs_0}^1 + \mu_{bs_0}^2}{2}$. Denoting again $\mathbb{1}_x$ the Dirac at x, we have

$$\mu_{bs_0}^1 = \frac{\mathbb{1}_{11} + \mathbb{1}_{\mathrm{j}2} + \mathbb{1}_{\mathrm{je}}}{3}, \quad \mu_{bs_0}^2 = \frac{\mathbb{1}_{\mathrm{j}1} + \mathbb{1}_{22} + \mathbb{1}_{\mathrm{je}}}{3}, \quad \text{and} \quad \mu_{bs_0} = \frac{\mathbb{1}_{11} + \mathbb{1}_{\mathrm{j}1} + \mathbb{1}_{\mathrm{j}2} + \mathbb{1}_{22} + 2\mathbb{1}_{\mathrm{je}}}{6}.$$

We claim that there is no policy $\boldsymbol{\delta}$ that induces distribution μ_{bs_0} on X_{bs_0}. Indeed, in a distribution induced by a policy $\boldsymbol{\delta}$, it follows from the parametrization that if $\mathbb{P}(X_a = 1) < 1$ and $\mathbb{P}(X_b = 1) > 0$, then $\mathbb{P}(X_b = 1, X_u = \mathrm{e}) > 0$. And, if $\mathbb{P}(X_a = 2) < 1$ and $\mathbb{P}(X_b = 2) > 0$, then $\mathbb{P}(X_b = 2, X_u = \mathrm{e}) > 0$. (In both claims, "if $\mathbb{P}(X_a = 1) < 1$" must be replaced by "if $\delta_a(x_{s_{k-1}}) \neq \mathbb{1}_i(x_{s_{k-1}})$" when $a = t_k$). As μ_{ub} is such that $\mathbb{P}(X_b = 1) > 0$, $\mathbb{P}(X_b = 2) > 0$, and $\mathbb{P}(X_b = 1, X_u = \mathrm{e}) = 0$, it cannot be induced by a policy. Hence, $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ is not convex. Therefore, $\mathcal{M}_{\mathcal{G}}(G, \boldsymbol{\rho})$ is not a polytope. □

9.2.3 Comparison of soluble and linear relaxations

MILP solvers are based on (much improved) branch-and-bound algorithms that use the linear relaxation to obtain bounds. Their ability to solve formulation (8.9) therefore depends on the quality of the bound provided by the linear relaxation. The following lemma ensures that adding arcs with head in the decision vertices gives an upper bound.

Lemma 9.8. *Let $G = (V, A)$ be an influence diagram with $V = V^{\mathrm{a}} \cup V^{\mathrm{a}}$. If $G' = (V, A')$ is such that A' is the union of A' and a set of arcs with head in V^{a}, then for every parametrization $\boldsymbol{\rho}$, $MEU(G', \boldsymbol{\rho})$ is a relaxation of $MEU(G, \boldsymbol{\rho})$.*

Proof. It is immediate since $\Delta_G \subseteq \Delta_{G'}$. □

Since SPU solves efficiently soluble influence diagrams, we could imagine alternative branch-and-bound schemes that use bounds computed using SPU on an influence diagram larger than G which is soluble. To formalize this idea, we introduce the following notion: A *soluble graph relaxation* of an influence diagram $G = (V, A)$ with $V = V^{\mathrm{s}} \cup V^{\mathrm{a}}$ is a soluble influence diagram $G' = (V, A')$ where A' is the union of A and a set of arcs with head in V^{a}. In this section, we show that the linear relaxation of MILP (8.9) is always at least as good as the ones obtained from soluble relaxations on our RJT.

Note that Theorem 8.12 can be reinterpreted as the link between soluble graph relaxation and linear relaxations. And since $\mathcal{M}(\overline{G}) = \overline{\mathcal{P}}$ and $\mathcal{M}(G^{\perp\perp}) = \mathcal{P}^{\perp\perp}$, by Theorem 9.5, $G^{\perp\perp}$ and $\overline{G}$ are soluble, and therefore soluble graph relaxations of G. Theorem 6.8 together with Corollary 8.2 ensure that being soluble is equivalent to the convexity of the set of achievable moments. Hence, a soluble relaxation corresponds to a convex relaxation of $\mathrm{MEU}(G, \boldsymbol{\rho})$.

Since any feasible strategy for the influence diagram G is a feasible strategy for a soluble graph relaxation G', for any parametrization $\boldsymbol{\rho}$, provides a bound on $\mathrm{MEU}(G, \boldsymbol{\rho})$. Since G' is soluble, this bound is tractable because it can be computed using the SPU algorithm. Soluble relaxations can therefore be used in branch-and-bound schemes for influence diagrams, as proposed in Khaled et al. [68]. To compare the relevance of such a scheme to our MILP approach we need to compare the quality of the soluble graph relaxation and linear relaxation bounds.

Corollary 9.9. *Let G' be a soluble graph relaxation of G, and $\mathcal{G}$ the RJT obtained by running Algorithm 5 on G'. Then the linear relaxation of* (8.9) *applied to G with RJT $\mathcal{G}$ provides a bound at least as good as the one provided by the soluble relaxation G'.*

Note that this bound can sometimes be strictly better thanks to constraints $(\boldsymbol{\mu}, \boldsymbol{\delta}) \in \mathcal{Q}^{\mathbf{b}}$.

Remark 14. In the literature, soluble relaxations have already been used to obtain bounds in different settings. For example Yuan et al. [166] used them in a branch and bound scheme. Their bounds rely on the notion of *sufficient information set* (SIS) for a decision vertex v [110].[2] SIS are set of vertices that have the following property: If, given an influence diagram G, we have a SIS D_v for each decision vertex v, then the influence diagram we obtain when we add arcs u, v for each u in D_v is a soluble relaxation of G. Different SIS may be available for a given decision vertex. Poh and Horvitz [121, Theorem 2] show that, the closer the SIS is to the descendant of the vertex, the worse the bound is; but the easier the inference is. Yuan et al. [166] make the choice of an easy inference and propose to use a SIS of minimum cardinality. An alternative option would be to add the perfect recall arcs, which would lead to a much harder inference problem but to better bounds. Using our $G^{\perp\!\!\perp}$ corresponds the following choice: among the SIS that enable to use our RJT for inference, use the one that leads to the best relaxation. $\triangle$

Proof of Corollary 9.9. Let $\mathcal{G}$ be the RJT obtained by running Algorithm 5 on G'. By Lemma 9.3, v is d-separated from $C_v \backslash \mathrm{fa}_{G'}(v)$ given $\mathrm{pa}_{G'}(v)$ in G'. Since G' is obtained by adding arcs to G, vertex v is therefore also d-separated from $C_v \backslash \mathrm{fa}_{G'}(v)$ given $\mathrm{pa}_{G'}(v)$ in G. We therefore obtain implies $A^{\perp\!\!\perp} \subseteq A'$. Thus, by Theorem 8.12, the bound provided by the linear relaxation of the MILP (8.9) is at least as good as the soluble graph relaxation bound. $\square$

9.3 Examples of non-soluble IDs solved by linear programs

As mentioned in Section 9.1, the fact that the linear relaxation of MILP (8.9) gives an optimal solution of $\mathrm{MEU}(G, \boldsymbol{\rho})$ is not a sufficient condition to be a soluble ID. In this section, we introduce some examples that can be solved by an optimal solution of the linear relaxation of MILP (8.9) and that are not soluble. In fact, we are able to characterize this class of influence diagrams, which is slightly larger than the soluble ones. However, describing this class of influence diagrams is beyond the scope of this thesis. In this section, we introduce an example of influence diagram that is not soluble and such that for every parametrization $\boldsymbol{\rho}$, the linear relaxation of MILP (8.9) gives an optimal solution of $\mathrm{MEU}(G, \boldsymbol{\rho})$. The numerical experiments in Section 9.5 illustrate that the SPU algorithm, whose the exactness characterizes the soluble IDs, badly performs on these examples.

Example 7. Consider a game with M players. The game is cooperative in the sense that all the players share the same goal. At each time t, all the players have access to the state of a game,

[2] When Nilsson and Höhle [110] wrote their paper, the notion of soluble influence diagram was still not known, and they used a weaker version. Their terminology is also different.

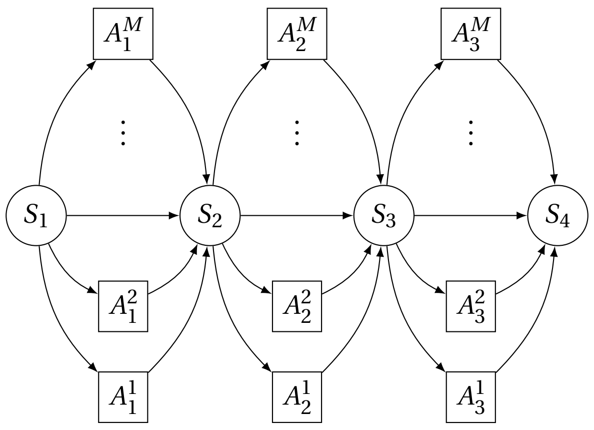


Figure 9.4 – Influence diagram of Example 7

denoted by S_t. Based on a state $S_t = s$, each player m in $[M]$ plays simultaneously an individual decision (or action) $A_t^m = a^m$ and the game's state transits randomly to state $S_{t+1} = s'$ according to transition probability $p(s'|s,\mathbf{a})$ where $\mathbf{a} = (a^m)_{m\in[M]}$ and this transition leads to an immediate reward $r(s,\mathbf{a},s')$. The choice of player m is modeled using individual policies $\boldsymbol{\delta}^m$ in Δ^m where Δ^m is the set of strategies of player m. A strategy is the vector of player policies, i.e., $\boldsymbol{\delta} = (\boldsymbol{\delta}^m)_{m\in[M]}$. We want to find the best policies for each player. Given a horizon T, the goal is to find a strategy $\boldsymbol{\delta}$ maximizing the total expected reward over the horizon T: $\max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}\big[\sum_{t=1}^T r(S_t,\mathbf{A}_t,S_{t+1})\big]$. Such a problem can be represented by the influence diagram in Figure 9.4. It can be checked that this influence diagram is non-soluble. $\triangle$

The reason why the SPU algorithm does not provide a strategy performing well on Example 7 is not clear. However, we believe that SPU algorithm fails when there are "parallel" decision vertices, i.e., pair of decision vertices such that there is no path on to each other. Indeed, the "parallel" decisions in Example 7 correspond to simultaneous decisions and the SPU algorithm optimizes the strategy by considering it sequentially.

9.4 Dual formulation for the soluble influence diagrams

The aim of this section is to prove Theorem 6.9, which we recall here:

Theorem 6.9. *The following properties hold:*

(i) *Linear program* (6.10) *is the dual of the linear relaxation of MILP* (6.6) *where variable* $\boldsymbol{\delta}$ *has been removed.*
(ii) *Linear program* (6.11) *is the dual of the linear relaxation of MILP* (6.6) *with valid inequalities* (6.7) *where variable* $\boldsymbol{\delta}$ *has been removed.*

Furthermore, the strong duality holds in both cases.

This theorem helps to understand how the vector of moments and the vector of value functions are related. While Theorem 6.7 ensures that there exists an RJT such that the linear relaxation of MILP (8.9) gives an optimal solution of $\mathrm{MEU}(G,\boldsymbol{\rho})$ for a soluble influence diagram, the following corollary ensures that this result can be extended to formulation (6.11).

Corollary 9.10. *If G is soluble, then there exists an RJT such that Linear program* (6.11) *induces an optimal solution of* $\mathrm{MEU}(G,\boldsymbol{\rho})$ *and both problems have the same optimal values. Such an RJT can be computed in polynomial time.*

Proof. Since G is soluble, there exists an RJT $\mathcal{G}$ such that $G^{\perp\!\!\!\perp} = G$. Theorem 6.9 ensures that the strong duality holds with the linear relaxation (8.9). Hence, by Theorem 6.7 the optimal value of the linear program (6.11) is equal to the optimal value of $\mathrm{MEU}(G,\boldsymbol{\rho})$. Now we build a feasible strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$ from an optimal solution of Linear program (6.11). We define $\boldsymbol{\delta}$ such that $\delta_v(x_v|x_{\mathrm{pa}(v)}) = 1$ when x_v belongs to $\arg\max_{x'_v} \sum_{x_{C_v^{\perp\!\!\!\perp}}} p_{C_v^{\perp\!\!\!\perp}|C_v^{\not\perp\!\!\!\perp}}(x_{C_v})\lambda_{C_v}(x'_v, x_{\check{C}_v})$ for any $x_{C_v^{\not\perp\!\!\!\perp}}$ and $v \in V^{\mathrm{a}}$. Since $C_v^{\not\perp\!\!\!\perp} = \mathrm{fa}(v)$, it follows that the δ_v depends on $x_{\mathrm{fa}(v)}$, ensuring that $\boldsymbol{\delta}$ is a feasible strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$. Since we consider an minimization problem, an optimal solution of Linear Program (6.11) satisfies for any vertex $v \in V$,

$$\begin{aligned}
\lambda_{C_v} &= \max_{\substack{x_u : u\in V^{\mathrm{a}},\\ C_u\in\mathrm{ch}(C_v)}} r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|\mathrm{fa}(C_u)}\lambda_{C_u}\\
&= r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \max_{\substack{x_u : u\in V^{\mathrm{a}},\\ C_u\in\mathrm{ch}(C_v)}} \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|\mathrm{fa}(u)}\lambda_{C_u}\\
&= r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}} \max_{x_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|\mathrm{fa}(u)}\lambda_{C_u}
\end{aligned}$$

ensuring that $\boldsymbol{\delta}$ reaches the optimal value of (6.11), which is the optimal value of $\mathrm{MEU}(G,\boldsymbol{\rho})$. □

Remark 15. Note that linear program (6.11) can be read as a linear formulation of the SPU algorithm on the corresponding RJT. If G is soluble, then the SPU algorithm builds an optimal strategy of $\mathrm{MEU}(G,\boldsymbol{\rho})$ by sequentially optimizing the individual policies on each decision vertex. The constraints of linear program (6.11) can be read as an iteration of the SPU algorithm by maximizing locally on each decision vertex $u \in V^{\mathrm{a}}$ as $\max_{x_u} \sum_{x_{C_u^{\perp\!\!\!\perp}}} p_{C_u^{\perp\!\!\!\perp}|\mathrm{fa}(u)}\lambda_{C_u}$. △

Proof of Theorem 6.9. We can remove the variables $\boldsymbol{\delta}$ from the linear relaxation of MILP (8.6) or MILP (8.9) because it does not play a role. Let $\mathcal{G} = (\mathcal{V},\mathcal{A})$ be an gradual RJT of G. We prove (i). We first reformulate the linear relaxation of MILP (8.6):

$$\max_{\boldsymbol{\mu}} \sum_{v\in V} \langle r_{C_v}, \mu_{C_v}\rangle \tag{9.3a}$$

$$\text{s.t. } \mu_{C_v} = p_{v|\mathrm{pa}(v)} \sum_{x_{\mathrm{pa}(C_v)\backslash C_v}} \mu_{\mathrm{pa}(C_v)} \; \forall v \in V^{\mathrm{s}} \tag{9.3b}$$

$$\sum_{x_v} \mu_{C_v} = \sum_{x_{\mathrm{pa}(C_v)\backslash C_v}} \mu_{\mathrm{pa}(C_v)} \qquad \forall v \in V^{\mathrm{a}} \tag{9.3c}$$

$$\boldsymbol{\mu} \geqslant 0 \tag{9.3d}$$

where the reward function r_{C_v} is defined in (6.4). This reformulation comes from the fact that the consistency constraints (6.5c) and the normalization constraints (6.5b) are induced by the constraints of (9.3). Now it remains to prove that the linear program (6.10) is the dual of the linear program (9.3). To do so, we compute the Lagrangian relaxation of Linear program (9.3). Let $\boldsymbol{\lambda} = (\lambda_{C_v}(x_{C_v}))_{x_{C_v}, v\in V}$ be the dual variables associated to the constraints (9.3b). Let $\boldsymbol{\pi} =$

$(\pi_{C_u\cap C_v}(x_{C_u\cap C_v}))_{x_{C_u\cap C_v},(C_u,C_v)\in\mathcal{A}}$ be the dual variables associated to the constraints (9.3c). Then, the Lagrangian is

$$
\begin{aligned}
&\mathcal{L}(\boldsymbol{\mu},\boldsymbol{\lambda},\boldsymbol{\pi})\\
&= \sum_{v\in V}\sum_{x_{C_v}} r_{C_v}(x_{C_v})\mu_{C_v}(x_{C_v}) + \sum_{v\in V^{\mathrm{s}}}\sum_{x_{C_v}}\lambda_{C_v}(x_{C_v})\left(\mu_{C_v}(x_{C_v}) - p_{v|\mathrm{pa}(v)}(x_{\mathrm{fa}(v)})\sum_{x_{\mathrm{pa}(C_v)\backslash C_v}}\mu_{\mathrm{pa}(C_v)}(x_{\mathrm{pa}(C_v)})\right)\\
&+ \sum_{v\in V^{\mathrm{a}}}\sum_{x_{\check{C}_v}}\pi_{C_v\cap\mathrm{pa}(C_v)}(x_{C_v\cap\mathrm{pa}(C_v)})\left(\sum_{x_v}\mu_{C_v}(x_{C_v}) - \sum_{x_{\mathrm{pa}(C_v)\backslash C_v}}\mu_{C_v}(x_{C_v})\right)\\
&= \sum_{v\in V^{\mathrm{s}}}\sum_{x_{C_v}}\mu_{C_v}(x_{C_v})\left(r_{C_v}(x_{C_v}) + \lambda_{C_v}(x_{C_v})\right)\\
&- \sum_{v\in V^{\mathrm{s}}}\sum_{x_{C_v}}\mu_{C_v}(x_{C_v})\left(\sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)})\lambda_{C_u}(x_{C_u}) + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u}(x_{C_u\cap C_v})\right)\\
&+ \sum_{v\in V^{\mathrm{a}}}\sum_{x_{C_v}}\mu_{C_v}(x_{C_v})\left(r_{C_v}(x_{C_v}) - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)})\lambda_{C_u}(x_{C_u}) - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u}(x_{C_u\cap C_v})\right.\\
&\left. + \pi_{\mathrm{pa}(C_v)\cap C_v}(x_{\mathrm{pa}(C_v)\cap C_v})\right)
\end{aligned}
$$

The dual problem of the linear program (9.3) is $\min_{\boldsymbol{\lambda},\boldsymbol{\pi}}\max_{\boldsymbol{\mu}\geqslant 0}\mathcal{L}(\boldsymbol{\mu},\boldsymbol{\lambda},\boldsymbol{\pi})$. Hence, the dual problem can be written as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & -\langle\lambda_{C_0}, p_0\rangle\\
\text{s.t.} \quad & r_{C_v} + \lambda_{C_v} - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u} \leqslant 0 \qquad \forall v\in V^{\mathrm{s}}\\
& r_{C_v} - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \pi_{\mathrm{pa}(C_v)\cap C_v} - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u} \leqslant 0 \quad \forall v\in V^{\mathrm{a}}
\end{aligned}
$$

This linear program can be reformulated as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \langle\lambda_{C_0}, p_0\rangle\\
\text{s.t.} \quad & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u} \qquad \forall v\in V^{\mathrm{s}}\\
& \pi_{\mathrm{pa}(C_v)\cap C_v} \geqslant r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u} \quad \forall v\in V^{\mathrm{a}}
\end{aligned}
$$

By introducing variables λ_{C_v} for $v\in V^{\mathrm{a}}$ such that

$$
\pi_{\mathrm{pa}(C_v)\cap C_v} \geqslant \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in\mathrm{ch}(C_v)}}\sum_{x_u} p_{u|\mathrm{pa}(u)}\lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in\mathrm{ch}(C_v)}}\pi_{C_v\cap C_u},
$$

we finally obtain the following formulation:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \langle \lambda_{C_0}, p_0 \rangle \\
\text{s.t. } & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \lambda_{C_u} \quad \forall v \in V
\end{aligned}$$

It achieves the proof since the last formulation corresponds exactly Linear Program (6.10).

Now we prove (ii). Like for the proof of (i), we reformulate the linear relaxation of MILP (8.9):

$$\max_{\boldsymbol{\mu}} \quad \sum_{v\in V} \langle r_{C_v}, \mu_{C_v} \rangle \tag{9.7a}$$

$$\text{s.t. } \mu_{C_v} = p_{v|\mathrm{pa}(v)} \sum_{x_{\mathrm{pa}(C_v)\backslash C_v}} \mu_{\mathrm{pa}(C_v)} \; \forall v \in V^{\mathrm{s}} \tag{9.7b}$$

$$\mu_{C_v} = p_{C_v^{\perp\!\!\perp}|C_v^{\not\perp\!\!\!\perp}} \sum_{x_{C_v^{\perp\!\!\perp}}} \mu_{C_v} \qquad \forall v \in V^{\mathrm{a}} \tag{9.7c}$$

$$\sum_{x_v} \mu_{C_v} = \sum_{x_{\mathrm{pa}(C_v)\backslash C_v}} \mu_{\mathrm{pa}(C_v)} \qquad \forall v \in V^{\mathrm{a}} \tag{9.7d}$$

$$\boldsymbol{\mu} \geqslant 0 \tag{9.7e}$$

Now it remains to prove that the linear program (6.11) is the dual of the linear program (9.7). Again, we compute the Lagrangian relaxation of linear program (9.7). Let $\boldsymbol{\lambda} = (\lambda_{C_v}(x_{C_v}))_{x_{C_v}, v\in V}$ be the dual variables associated to the constraints (9.7b) and constraints (9.7c). Let $\boldsymbol{\pi} = (\pi_{C_u\cap C_v}(x_{C_u\cap C_v}))_{x_{C_u\cap C_v},(C_u,C_v)\in\mathcal{A}}$ be the dual variables associated to the constraints (9.7d). Using the previous calculus, the Lagrangian $\mathcal{L}$ can be written:

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\mu},\boldsymbol{\lambda},\boldsymbol{\pi}) \\
&= \sum_{v\in V^{\mathrm{s}}} \sum_{x_{C_v}} \mu_{C_v}(x_{C_v}) \Bigg(r_{C_v}(x_{C_v}) + \lambda_{C_v}(x_{C_v}) - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) \lambda_{C_u}(x_{C_u}) \Bigg) \\
&- \sum_{v\in V^{\mathrm{s}}} \sum_{x_{C_v}} \mu_{C_v}(x_{C_v}) \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \pi_{C_v\cap C_u}(x_{C_u\cap C_v}) \\
&+ \sum_{v\in V^{\mathrm{a}}} \sum_{x_{C_v}} \mu_{C_v}(x_{C_v}) \Bigg(r_{C_v}(x_{C_v}) - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)}(x_{\mathrm{fa}(u)}) \lambda_{C_u}(x_{C_u}) - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \pi_{C_v\cap C_u}(x_{C_u\cap C_v}) \\
&+ \pi_{\mathrm{pa}(C_v)\cap C_v}(x_{\mathrm{pa}(C_v)\cap C_v}) + \lambda_{C_v}(x_{C_v}) - \sum_{x_{C_v^{\perp\!\!\perp}}} p_{C_v^{\perp\!\!\perp}|C_v^{\not\perp\!\!\!\perp}}(x_{\check{C}_v}) \lambda_{C_v}(x_{C_v}) \Bigg)
\end{aligned}$$

Hence, the dual problem is the following:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & -\langle \lambda_{C_0}, p_0 \rangle \\
\text{s.t. } & r_{C_v} + \lambda_{C_v} - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \pi_{C_v\cap C_u} \leqslant 0 \qquad \forall v \in V^{\mathrm{s}} \\
& r_{C_v} + \lambda_{C_v} - \sum_{x_{C_v^{\perp\!\!\perp}}} p_{C_v^{\perp\!\!\perp}|C_v^{\not\perp\!\!\!\perp}} \lambda_{C_v} - \sum_{\substack{u\in V^{\mathrm{s}}:\\ C_u\in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \pi_{\mathrm{pa}(C_v)\cap C_v} - \sum_{\substack{u\in V^{\mathrm{a}}:\\ C_u\in \mathrm{ch}(C_v)}} \pi_{C_v\cap C_u} \leqslant 0 \forall v \in V^{\mathrm{a}}
\end{aligned}$$

This linear program leads to the following one:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \langle \lambda_{C_0}, p_0 \rangle \\
\text{s.t.} \quad & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \pi_{C_v \cap C_u} && \forall v \in V^{\mathrm{s}} \\
& \pi_{\mathrm{pa}(C_v) \cap C_v} + \lambda_{C_v} - \sum_{x_{C_v^{\perp\!\!\perp}}} p_{C_v^{\perp\!\!\perp} | C_v^{\not\perp\!\!\perp}} \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \pi_{C_v \cap C_u} && \forall v \in V^{\mathrm{a}}
\end{aligned}$$

For any optimal solution of Linear Program (9.10), we have:

$$\begin{aligned}
\pi_{\mathrm{pa}(C_v) \cap C_v} &\geqslant \sum_{x_{C_v^{\perp\!\!\perp}}} p_{C_v^{\perp\!\!\perp} | C_v^{\not\perp\!\!\perp}} \lambda_{C_v} \\
\lambda_{C_v} &\geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \pi_{C_v \cap C_u}
\end{aligned}$$

Indeed, otherwise it can be showed that we can obtain an optimal solution with a lower value. Hence it follows that linear program (9.10) becomes

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \langle \lambda_{C_0}, p_0 \rangle \\
\text{s.t.} \quad & \lambda_{C_v} \geqslant r_{C_v} + \sum_{\substack{u \in V^{\mathrm{s}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_u} p_{u|\mathrm{pa}(u)} \lambda_{C_u} + \sum_{\substack{u \in V^{\mathrm{a}}: \\ C_u \in \mathrm{ch}(C_v)}} \sum_{x_{C_u^{\perp\!\!\perp}}} p_{C_u^{\perp\!\!\perp} | C_u^{\not\perp\!\!\perp}} \lambda_{C_u} && \forall v \in V
\end{aligned}$$

It proves (ii). The strong duality holds in both case because there always exist an optimal solution of the primal problems (e.g. [93]). It achieves the proof. □

9.5 Numerical experiments

In this section, we give numerical results on random instances of Example 7. As mentioned in section 9.3, this example is represented by an influence diagram that is non-soluble and such that an optimal solution of MEU$(G, \boldsymbol{\rho})$ can be found by solving the linear relaxation of MILP (8.9). All linear programs have been implemented in Julia with JuMP interface [41] and solved using `Gurobi` 9.0 [52]. Experiments have been run on a server with 192Gb of RAM and 32 cores at 3.30GHz.

The instances are generated by first choosing $\mathrm{k}_s = |\mathcal{X}_S|$ and $\mathrm{k}_a = |\mathcal{X}_A^1| = \cdots = |\mathcal{X}_A^M|$. We then randomly generate the initial probability distributions $\big(p(s)\big)_{s \in \mathcal{X}_S}$, the transition probability distributions $\big(p(s'|s, a^1, \ldots, a^M)\big)_{\substack{s \in \mathcal{X}_S \\ a^1 \in \mathcal{X}_A^1, \ldots, a^M \in \mathcal{X}_A^M}}$, and the immediate reward functions $\big(r(s, a^1, \ldots, a^M, s')\big)_{\substack{s \in \mathcal{X}_S \\ a^1 \in \mathcal{X}_A^1, \ldots, a^M \in \mathcal{X}_A^M}}$. An instance is the tuple $(M, \mathrm{k}_s, \mathrm{k}_a, \mathfrak{p}, \mathbf{r})$. We choose $(M, \mathrm{k}_s, \mathrm{k}_a) \in \{3, 5\}^3$ and we generate 50 instances $(M, \mathrm{k}_s, \mathrm{k}_a, \mathfrak{p}, \mathbf{r})$.

Metrics. We denote respectively by z^* and z_{R}^* the optimal values of MEU$(G, \boldsymbol{\rho})$, which is computed by solving MILP (8.9), and the optimal value of the linear relaxation of MILP (8.9). We compare these values with the value z^{SPU} obtained by running the SPU algorithm. To do so, for

$(M, \mathrm{k}_s, \mathrm{k}_a)$	T	$\lvert\Delta^{\mathrm{d}}_{\mathrm{ml}}\rvert$	Alg.	Gap (%)	Time (s)
(3,3,3)	10	10^{42}	MILP (8.9)	Opt.	0.05
			SPU	−5.08	7.96
			Lin. Relax. MILP (8.9)	0.00	0.05
	20	10^{84}	MILP (8.9)	Opt.	0.12
			SPU	−5.09	15.92
			Lin. Relax. MILP (8.9)	0.00	0.10
(3,5,5)	10	10^{84}	MILP (8.9)	Opt.	0.24
			SPU	−7.06	8.42
			Lin. Relax. MILP (8.9)	0.00	0.28
	20	10^{168}	MILP (8.9)	Opt.	0.62
			SPU	−7.06	16.84
			Lin. Relax. MILP (8.9)	0.00	0.57
(5,3,3)	10	10^{71}	MILP (8.9)	Opt.	2.25
			SPU	−11.38	7.59
			Lin. Relax. MILP (8.9)	0.00	1.07
	20	10^{142}	MILP (8.9)	Opt.	7.52
			SPU	−11.37	15.17
			Lin. Relax. MILP (8.9)	0.00	2.92
(5,5,5)	10	10^{174}	MILP (8.9)	Opt.	29.67
			SPU	−13.71	7.85
			Lin. Relax. MILP (8.9)	0.00	16.89
	20	10^{348}	MILP (8.9)	Opt.	1374.61
			SPU	−13.75	15.68
			Lin. Relax. MILP (8.9)	0.00	1311.62

Table 9.1 – Average results on 50 instances of the cooperative game.

each value $z \in \{z^{\mathrm{SPU}}, z^*_{\mathrm{R}}\}$ we compute the relative gaps with respect to the value of optimal value $g(z) = \frac{z-z^*}{z^*}$ and we report it as well as the computation time of each mathematical program in Table 9.1. All the mathematical programs have been solved optimally. The results are averaged over the set 50 generated instances and reported in Table 9.1 The first column indicates the value of the triplet $(M, \mathrm{k}_s, \mathrm{k}_a)$. The second column indicates the value of the finite horizon T. The third column indicates the size of the set of deterministic strategies $\lvert\Delta^{\mathrm{d}}_{\mathrm{ml}}\rvert$. Finally, the last three columns indicate the algorithms used (Alg.), the average gap value over the 50 instances, and the averaged computation time.

The results in Table 9.1 show that the linear relaxation of MILP (8.9) gives the same optimal value as that of MILP (8.9). It validates the fact that there are non-soluble influence diagrams for which the linear relaxation of MILP (8.9) gives an optimal strategy. Since the SPU strategy is a local optimum, the value of the gap $\frac{z^{\mathrm{SPU}}-z^*}{z^*}$ is non-positive as shown in Table 9.1. One can also observe that, as we expected, the SPU algorithm gives a strategy with an objective value which can be far from the optimal value. In particular these gaps are significantly larger than those of the numerical experiments in Section 8.6. Note that when the number of players increases, the gap $g(z^{\mathrm{SPU}})$ decreases also. It supports our intuition that when the influence diagram has "parallel" decision vertices, the SPU algorithm fails to give a good strategy.

Maintenance problem at Air France Part III

10 Data-driven maintenance optimization

In this chapter, we focus on the airplane maintenance problem at Air France, which we recall here. Given an airplane planning with scheduled maintenance slots, the decision maker receives sensor data at each maintenance slot on M equipments of an airplane and chooses at most K equipments that should be maintained during this maintenance slot. The objective is to choose a maintenance policy maximizing the expected costs, which correspond to the sum of the maintenance costs and the failure costs.

The sensor data correspond to a collection of time series recorded at 1Hz during flights. In Chapter 3, we introduced a generic predictive maintenance problem and we formalized it as a weakly coupled POMDP problem to build a policy for the maintenance of the components of a system. We wish to use such a maintenance policy in the airplane maintenance problem at Air France. This requires to cast the airplane maintenance problem as an instance of the generic predictive maintenance problem described in Chapter 3. However, this raises three practical issues. First, in the airplane maintenance problem, at each maintenance slot the decision maker has access to sensor data, which are continuous and high dimensional, instead of discrete observations. Consequently, it requires to create a discretization method that transforms the sensor data recorded during a flight into a discrete observation for each equipment. Second, Air France's requirement is that the resulting discrete observations be interpretable. Third, the parameters of the POMDPs, which compose the weakly coupled POMDP, are not available in practice. This requires to estimate these parameters. In this chapter, we describe a statistical methodology that addresses these three issues and that casts the airplane maintenance problem as a weakly coupled POMDP problem. This chapter is organized as follows:

- Section 10.1 describes the current Air France's approach and the main steps of our approach to address the three issues mentioned above. In particular, we argue about our choices of statistical tools.
- Section 10.2 formally introduces the problem of finding a maintenance policy given sensor data.
- Section 10.3 describes how we transform such a maintenance problem into a weakly coupled POMDP. In particular, we detail how we estimate the parameters of the weakly coupled POMDP.
- Section 10.4 provides numerical experiments on a simulated system on which we apply the policy of Chapter 5, which is a feasible policy of the weakly coupled POMDP. We com-

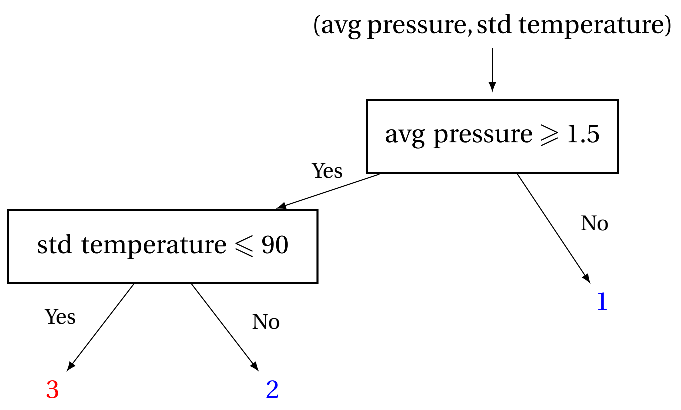


Figure 10.1 – An example of a decision tree that takes as inputs the average (avg) pressure and the standard deviation (std) of the temperature. It returns a discrete label in $\{1,2,3\}$. Each label corresponds to a cluster. Clusters 1 and 2 correspond to normal behavior (blue), and cluster 3 corresponds to high-failure risk (red).

pare the results of this policy against a maintenance policy reproducing Air France's one on the simulated system. Although we are not able to simulate the airplane's sensor data, we also provide numerical experiments on Air France's dataset by comparing the past failures and what our maintenance policy would have suggested.

- Section 10.5 contains bibliographical remarks.

10.1 About the airplane maintenance problem at Air France

In this section, we describe the current approach at Air France to address the airplane maintenance problem and the main steps of our approach. We recall that we have access to a dataset containing sensor data that correspond to a collection of time series over several years on the whole fleet of airplanes.

Current approach at Air France and its limits. Air France already uses predictive maintenance for a few failure prone equipments. In their current practice, the maintenance engineers use *fault trees* to support their decisions. Using machine learning terminology, fault trees are equivalent to *decision trees*, generally hand-designed and of small size. In this dissertation, we will always designate the tool used in Air France as decision tree. A decision tree is illustrated on Figure 10.1. It takes in input a vector of features extracting relevant information from the time series. Using a succession of binary rules splitting the features space in two, it partitions this feature space into a small number of clusters. Each of the clusters is informative from an engineering point of view: some correspond to normal behavior (blue labels in Figure 10.1), some to high failure risk (red labels in Figure 10.1). The engineers then maintain the equipment if the decision tree returns a label corresponding to a high failure risk cluster. This approach is a *diagnosis*-based [3] heuristic and is blind about the failure risk for each equipment over the remaining of the horizon. It is easy to understand why this approach enabled to drastically reduce the failures and the maintenance costs of the small number of failure-prone equipments considered. Indeed, when only very few equipments are considered, prioritizing between equipments is not an issue, and this heuristic makes perfect sense. In addition, even if the decision trees usually provide a diagnosis, Air France designs their decision tree in such a way that it detects the first symptoms of deterioration. This characteristic ensures that the heuristic is more preventive than corrective, i.e., an equipment is maintained before it fails. Furthermore, the decision tree used can make these diagnosis very accurately because it leverages a physical understanding of the equipment.

However, such an approach cannot be extended to a large number of equipments. The first reason is mathematical. When there are many equipments and scarce maintenance resources, anticipating the future on several decision steps becomes crucial, and a diagnosis-based heuristic cannot work anymore. We therefore need a richer multistage stochastic optimization approach, which itself requires a richer prediction model. The second reason is industrial: Building manually a good decision tree requires several months of work of an expert. Given that experts are also a scarce resource, such an investment cannot be scaled to dozens of equipments, and we therefore need to use prediction models that require less expert time.

How to build a prediction model trusted by maintenance engineers. There are two ways of building trust in models: experimental testing, or validation by experts who understand the model. A specific difficulty comes from the fact that the data is *censored*. Indeed, taking maintenance decisions requires to be able to predict the behavior of the system just before it fails. But since failures are costly, airlines try to avoid them as much as possible. If we have much data on the system when it works well, we have a small amount of data on the behavior of the system right before it fails: On the whole history, the number of failures observed on one equipment never exceeds 10. Given the small number of failures in our dataset, and that we do not have access to a simulator, experimental validation is not possible. *We must therefore propose a model that maintenance experts can validate, but whose design and validation do not require too much of their time.* We solve that conundrum as follows: *our model makes decisions using only information that maintenance experts can easily understand*. Since decision trees form the standard method used by the engineers of Air France to take maintenance decisions, we interpret “information that maintenance experts can easily understand” as “the result of a small size decision tree.”[1] To reduce our need of expert time, the decision trees are learned from historical data, and experts only check that the classification they produce makes sense from an engineering point of view. An additional difficulty to build a decision tree given our dataset is that the data are not labeled. Since we do not know when an equipment exactly fails, we are not able to assign a label on each flight indicating if the equipment has an abnormal behavior due to a failure.

Main steps of our approach. The goal of our approach is to cast the airplane maintenance problem as a weakly coupled POMDP and the decision maker chooses which equipments to maintain at each maintenance slot according to a feasible policy of this weakly coupled POMDP. To do so, we proceed in several steps which are summarized in Figure 10.2. We first extract for each monitored equipment a vector of features in $\mathbb{R}^d$ from the sensor data. We then use a de-

[1] Since decision trees are widely used for maintenance across several industries [26, 131, 150], this assumption is not specific to Air France.

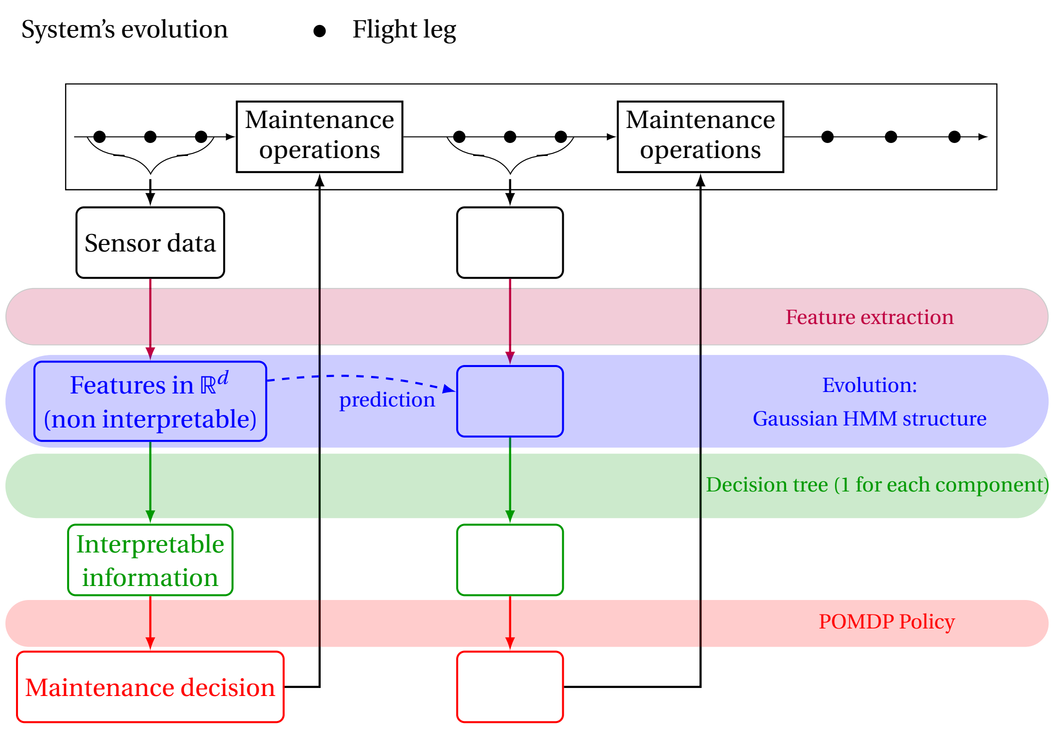


Figure 10.2 – The four elements of our approach: the feature extraction (in purple), the prediction model (in blue), the decision tree (in green) and the policy (in red).

cision tree to turn this vector of features into a label that can be interpreted by experts. Finally, based on the labels of all equipments, our policy chooses which equipments should be maintained in such a way that the number of equipments to maintain does not exceed the maintenance capacity. This maintenance is then performed, and the airplane operates flights until the next maintenance slot. Using the discrete labels as observations at each maintenance slot, the resulting problem is a predictive maintenance problem as the one described in Chapter 3.

Learning to predict the evolution of sensor data would make little sense since it is very noisy and high dimensional. We therefore learn a *Gaussian Hidden Markov model* (HMM) that predicts the evolution of the vector of features. Learning such a statistic model consists in estimating its parameters, which are the transition probability distributions and the Gaussian law parameters. We must specify how we extract the features, how we learn the Gaussian HMM parameters and the decision tree that transforms the Gaussian HMM into a HMM with discrete observations. If the way we extract features and learn the Gaussian HMM parameters is relatively standard, the way we learn the decision tree is less so. Indeed, in predictive maintenance the decision trees are generally hand-designed for maintenance diagnosis, and not automatically learned to provide the input of a policy. The Gaussian HMM parameters together with the learned decision tree lead to the parameters of a HMM with discrete observations, which are the outputs of the decision tree. These HMM parameters are learned for each equipment, which is sufficient to define the weakly coupled POMDP parameters.

Once the weakly coupled POMDP parameter have been set, we can use the maintenance policy proposed in Part I. It raises a practical issue: Evaluating the performance of our approach requires to evaluate the policy it returns on the true system, and not only according to the model we learned. Since we do not have a simulator for the sensor data of the airplanes equipments, we evaluate the performance of our approach using a simple simulator of a system with multiple deteriorating components. To evaluate our maintenance policy on the airplane maintenance problem, we compare the maintenance times in the historical dataset against what our maintenance policy would have done.

We emphasize that all the steps of our approach are relatively easy to implement.

10.2 Formalizing the airplane maintenance problem

We consider a system on a horizon $\mathcal{T} \in \mathbb{Z}_+$. This system is composed of M equipments indexed by $m \in [M]$. These equipments are subject to failures, and for $\tau \in [\mathcal{T}]$, we denote by F_τ^m the binary random variable equal to 1 if a failure happens on equipment m at date τ. The sensor signals are recorded, and we denote by Z_τ^m the signals in $\mathbb{R}^k \times \{0,1\}^{k'}$ with $k, k' \in \mathbb{N}$, recorded on equipment m at date $\tau \in [\mathcal{T}]$. There are T maintenance slots scheduled on given (deterministic) dates τ_t, with $\tau_t < \tau_{t+1}$ for every $t \in [T]$. On each maintenance slot, the decision maker chooses to maintain at most K equipments. We denote by A_t^m the binary random variable equal to 1 if equipment m is maintained on slot t. At each maintenance slot, at most K equipments can be maintained. Hence, the action $\big(A_t^m\big)_{m\in[M]}$ on slot t belongs to the action space $\mathcal{X}_A$ defines as follows

$$\mathcal{X}_A = \left\{ \mathbf{a} = (a^m)_{m\in[M]} \in \{0,1\}^M : \sum_{m\in[M]} a^m \leqslant K \right\}. \tag{10.1}$$

Maintaining equipment m costs c_{m}^m, while a failure on this equipment costs c_{f}^m. A *maintenance policy* $\mathfrak{d}$ is a conditional distribution of $\big(A_t^m\big)_{m\in[M]}$ given $\Big((Z_\tau^m, F_\tau^m)_{\tau\leqslant\tau_t, m\in[M]}, (A_{t'}^m)_{t'<t, m\in[M]}\Big)$. We denote by $\mathfrak{D}$ the set of maintenance policies. The objective is to find a policy that minimizes the expected cost

$$\min_{\mathfrak{d}\in\mathfrak{D}} \mathbb{E}\left(\sum_{m=1}^{M} \left(c_{\mathrm{f}}^m \sum_{\tau=1}^{\mathcal{T}} F_\tau^m + c_{\mathrm{m}}^m \sum_{t=1}^{T} A_t^m \right) \middle| \mathfrak{d} \right). \tag{10.2}$$

The approach is *data-driven*: we do not have access to any model of the system, nor to any simulator of the component, and we do not know if sensor signals provide enough information to model the dynamic of the system. In particular, we do not have access to the probability distribution on $(Z_\tau^m)_{m\in[M]}$, $(F_\tau^m)_{m\in[M]}$, and $(A_t^m)_{m\in[M]}$, and therefore have no way to evaluate the objective function of (10.2). We only have access to historical values taken by these random variables on a previous horizon. As mentioned in Section 10.1, the historical dataset contains a small number of failures. Hence, the probability distributions cannot be learned precisely. Furthermore, since we cannot evaluate a policy using a simulator, such a policy must be interpretable. Since Problem (10.2) is hard to solve, we propose to restrict the set of policies.

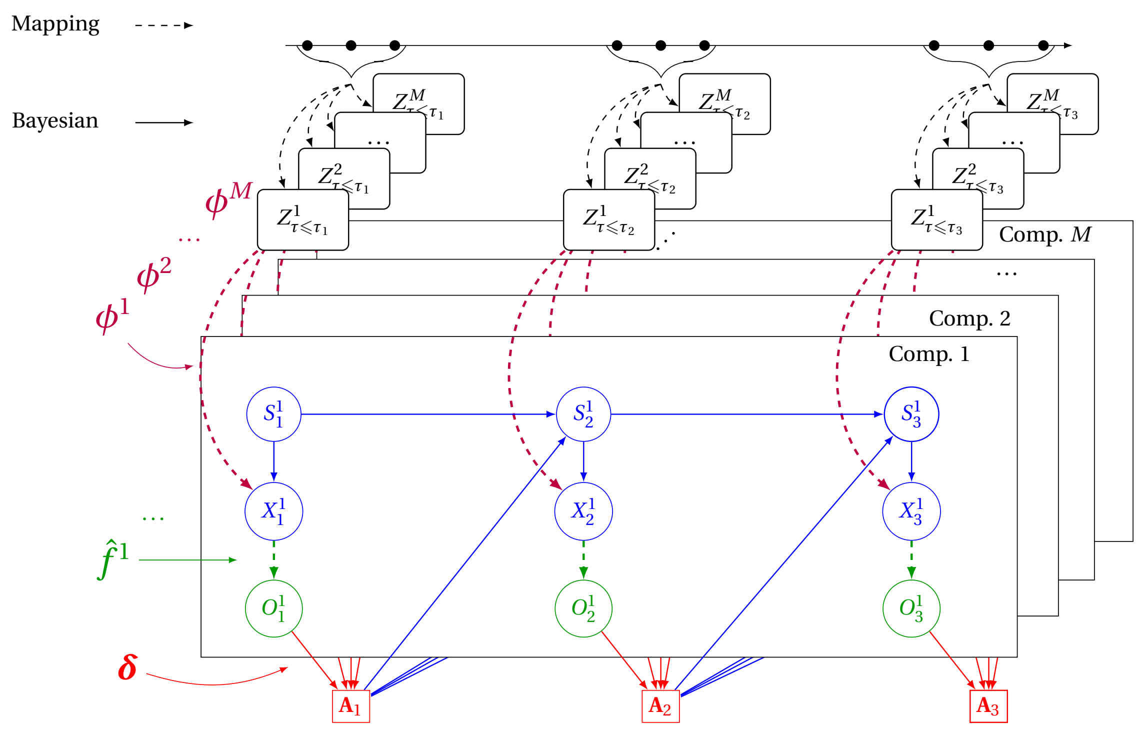

Mapping
Bayesian
$Z^M_{\tau\leqslant\tau_1}$
$Z^2_{\tau\leqslant\tau_1}$
$Z^1_{\tau\leqslant\tau_1}$
$Z^M_{\tau\leqslant\tau_2}$
$Z^2_{\tau\leqslant\tau_2}$
$Z^1_{\tau\leqslant\tau_2}$
$Z^M_{\tau\leqslant\tau_3}$
$Z^2_{\tau\leqslant\tau_3}$
$Z^1_{\tau\leqslant\tau_3}$
ϕ^M
ϕ^2
ϕ^1
Comp. M
Comp. 2
Comp. 1
S^1_1
S^1_2
S^1_3
X^1_1
X^1_2
X^1_3
O^1_1
O^1_2
O^1_3
$\hat{f}^1$
δ
A_1
A_2
A_3

Figure 10.3 – A graphical representation of our approach using the notation introduced in Section 10.2. First, the feature extraction $\boldsymbol{\phi}$ from sensor data is represented by purple dashed arcs. Second, the probabilistic dependences of our Gaussian HMMs are represented by blue plain arcs for each equipment. Third, the mapping of our decision tree $\hat{\boldsymbol{f}}$ is represented by dotted green arcs. Fourth, the policy of the decision maker is represented by red plain arcs. Dashed arcs indicate a mapping. Plain arcs indicate the probabilistic dependences as for a Bayesian network.

10.3 Modeling as a weakly coupled POMDP

In this section, we explain how to model the general maintenance problem (10.2) as a weakly coupled POMDP and how to learn the weakly coupled POMDP parameters. Hence, using notation of 3, we have to set

$$\text{the value of } \mathcal{X}^m_S, \mathcal{X}^m_O, \mathcal{X}^m_A, \mathfrak{p}^m, \mathbf{r}^m \text{ for } m \in [M], \text{ and the value of } \mathcal{X}_A. \tag{10.3}$$

In the usual POMDP methodology, the observations O^m_t are the outputs of the system directly observed by the decision maker on the system, and the POMDP parameters $\mathfrak{p}$ are learned from a history of trajectories on O^m_t, A^m_t on each equipment m. On the contrary, in the content of Problem (10.2), the decision maker observes time series Z^m_τ in $\mathbb{R}^{k^m} \times \{0,1\}^{k'^m}$ and not observations in a finite set $\mathcal{X}^m_O$. In order to turn the sensor data into observations in a finite set $\mathcal{X}^m_O$, we use two successive mappings ϕ^m and $\hat{f}^m$ as follows.

$$\begin{tikzcd}
(\mathbb{R}^{k^m} \times \{0,1\}^{k'^m})^{[0,\tau_t]} \arrow[r, "\phi^m"] & \mathbb{R}^{d^m} \arrow[r, "\hat{f}^m"] & \mathcal{X}_O^m \\
(z_\tau^m)_{\tau \leqslant \tau_t} \arrow[r, mapsto] & x_t^m \arrow[r, mapsto] & o_t^m
\end{tikzcd}$$

First, we extract features X_t^m in $\mathcal{X}_X^m := \mathbb{R}^{d^m}$ from the times series $(Z_\tau^m)_{\tau \leqslant \tau_t}$ using a manually defined ϕ (represented with dashed purple arcs in Figure 10.3). Second, we turn these features into observations in a finite set $\mathcal{X}_O^m$ using a decision tree $\hat{f}^m$ (represented with dashed green arcs on Figure 10.3) that we learn from the data. We will explain later how we define $\mathcal{X}_O^m$. We define the observation at time t as $O_t^m := \big(\hat{f}^m \circ \phi\big)\big((Z_\tau^m)_{\tau \leqslant \tau_t}\big)$. To obtain a policy $\mathfrak{d}$ for Problem (10.2), we then proceed as follows. We learn the parameters θ^m of a weakly coupled POMDP based on the history of the observations, and we compute an approximate policy $\boldsymbol{\delta}$ for the weakly coupled POMDP (see Chapter 5), and then retrieve a maintenance policy $\mathfrak{d}$ for the initial problem (10.2) using

$$\mathfrak{d} := \boldsymbol{\delta} \circ \hat{\mathbf{f}} \circ \boldsymbol{\phi} \tag{10.4}$$

where $\hat{\mathbf{f}} = (\hat{f}^1, \ldots, \hat{f}^M)$ and $\boldsymbol{\phi} = (\phi^1, \ldots, \phi^M)$. In the remaining of this section, we explain how to choose ϕ^m, and how to learn $\hat{f}^m$, and finally how to set the weakly coupled POMDP parameters (10.3).

Choice of features vector ϕ^m. We take in input the sensor signals time series that have been selected by the maintenance experts as the most relevant ones. We then compute a moderate number of features standardly used in predictive maintenance such as peak values, mean, standard deviation, etc [63, 153]. We then select the most relevant features using the maintenance expert knowledge. We end up with 10 to 50 features per equipment. Note that due to the large amount of historical sensor data manipulated (typically for one flight and one equipment there are 20 time series each containing 20000 data points), big data technologies must be used. We use `Spark` [167].

Learning $\hat{f}$ and the POMDP is more challenging, as we now detail.

Learning the decision tree $\hat{f}^m$. We introduce a methodology to learn $\hat{f}^m$, in such a way that the partition of the feature space $\mathbb{R}^{d^m}$ it realizes is informative about the dynamic of the system. Our methodology is in two steps.

1. *Learning HMM to predict the evolution of the feature vector X_t^m.* We compute $X_t^m = \phi^m\big((R_\tau^m)_{\tau \leqslant t}\big)$ on our learning dataset to obtain the trajectories followed by the different features, and then we learn a Gaussian HMM (represented with blue arcs in Figure 10.2) with n_S^m hidden states to predict the evolution of the features X_t^m using standard algorithms. The number of states n_S^m is chosen carefully.
2. *Learning the decision tree $\hat{f}^m$.* On our learning dataset, we recompute the most probable hidden state $\hat{S}_t{}^m$ according to the HMM learned at the previous step, and we choose $\hat{f}^m$ as follows: We train a decision tree to predict the most probable state $\hat{S}_t^m$ given X_t^m. Since we have a dataset of labeled data $(X_t^m, \hat{S}_t^m)$, we do this by learning a decision tree using standard CART algorithms [19].

We now provide details on each of these steps.

1. Learning the Gaussian HMM that models the features evolution. Following well-established practice in the maintenance literature [51, 82] we use a Gaussian left-right HMM. Kim et al. [73] show that Gaussian models are appropriate to predictive maintenance. Denote by $\tilde{p}^m$ the transition probability distribution of the Gaussian HMM of equipment m. We choose a number of hidden states n_S^m, and we set $\mathcal{X}_S^m := [n_S^m]$. The hidden state corresponds to the degradation level. A *left-right HMM* is a HMM such that there exists an total ordering $\prec$ on $\mathcal{X}_S^m$ and for all $s, s' \in \mathcal{X}_S^m$, $\tilde{p}^m(s'|s) = 0$ when $s' \prec s$. Left-right HMMs [125, Fig. 6] enable to model deteriorating systems, since the assumptions on the transition probability distribution enable to model the fact that a equipment cannot repair itself. We assume that the *failure state* s_F^m is the maximal element with respect to order $\prec$. Following the literature, we use the Baum-Welch Algorithm [11] to learn the HMM parameters.

Remark 16. The number of hidden states n_S^m is also a parameter. Like Le et al. [82], we choose the value of n_S^m that leads to the minimal value of the Bayes Information Criterion (BIC) [53, Sec 7.7]. △

2. Learning the decision tree $\hat{f}^m$. Learning algorithms for decision trees require labeled data input. We describe here how we label a sequence of features in $\mathbb{R}^{d^m}$, $x_1, x_2, \dots, x_N$. We use our Gaussian HMM to predict the most probable sequence of hidden states $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_T$ using the Viterbi algorithm [155]. This gives us a label $\hat{s}_t$ to data point x_t, for all t in $[T]$. Finally, we learn a decision tree that predicts the value of the hidden state $\hat{s}_t$ given the observation x_t. We choose to learn such a decision tree because each state in $\mathcal{X}_S^m$ corresponds to a degradation state of the equipment. Hence, our decision tree predicts the deterioration of the equipment. Let $\hat{f}^m$ be the learned decision tree. It maps any continuous features to discrete states, i.e., $\hat{f}^m : \mathbb{R}^{d^m} \to \mathcal{X}_O^m$ where $\mathcal{X}_O^m := \mathcal{X}_S^m$ is a finite space. We choose to use the CART (Classification And Regression Trees) algorithm [19] to learn $\hat{f}^m$. Numerical experiments show that the learned decision tree has a satisfying accuracy.

Having estimated a Gaussian HMM and a decision tree for each equipment, we finally describe how we set the weakly coupled POMDP parameters (10.3).

Setting the weakly coupled POMDP parameters (10.3)**.** We already have defined $\mathcal{X}_S^m$ and $\mathcal{X}_O^m$ on each equipment m. We use the action space $\mathcal{X}_A \subseteq \{0,1\}^M$ defined in (10.1), which has the form (3.2) by setting $\mathbf{D}^m(a) = a$ for all a in $\{0,1\}$ and $\mathbf{b} = 1$.

Now we set the parameters $\mathfrak{p}^m$ for any equipment m. From step 1, we already have the probability distributions $\mathbb{P}(S_1)$, $\mathbb{P}(S_{t+1}|S_t)$ and the Gaussian law parameters of the emission probability distributions. From step 2, we already have the learned decision tree $\hat{f}$. We set the following probability distributions.

$$p^m(s'|s,a) := \mathbb{P}\big(S^m_{t+1} = s'|S^m_t = s, A^m_t = a\big) = \begin{cases} \tilde{p}^m(s) & \text{if } a = 1 \\ \tilde{p}^m(s'|s) & \text{otherwise} \end{cases} \tag{10.5}$$

$$p^m(o|s) := \mathbb{P}\big(O^m_t = p|S^m_t = s\big) = \mathbb{E}\Big[\mathbb{1}_{\{\hat{f}(X^m_t)=o\}}|S^m_t = s\Big] \tag{10.6}$$

for all $s, s' \in \mathcal{X}^m_S$, $o \in \mathcal{X}^m_O$, $a \in \{0,1\}$ and $m \in [M]$. Note that we assumed in (10.5) that when a equipment has been maintained, the equipment is new. The right-hand side of (10.6) is an integral over $\mathbb{R}^{d^m}$, which is difficult to compute since the vector of features is not independent. Therefore, we compute it using Monte-Carlo simulation.

Now we set the reward function. We associate a maintenance cost c^m_{m} and a failure cost c^m_{f} to each equipment m. The individual immediate reward function can be written

$$r^m(s,a,s') = -\mathbb{1}_{s'=s^m_F} c^m_{\mathrm{f}} - \mathbb{1}_{a=1} c^m_{\mathrm{m}},$$

for all $s, s' \in \mathcal{X}^m_S$, $a \in \{0,1\}$ and $m \in [M]$.

We solve Problem (10.2) using a maintenance policy $\mathfrak{d} := \boldsymbol{\delta} \circ \hat{\mathbf{f}} \circ \boldsymbol{\phi}$, where $\boldsymbol{\delta}$ is a solution of $\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$ (see Section 3.2).

Discussion. When we cast Problem (10.2) as a weakly coupled POMDP we restrict ourselves to maintenance policies of the form (10.4). This restriction enables to address several challenges.

First, due to the large dimension of the sensor data recorded on each flight we are not able to manipulate it in machine learning algorithms. Hence, it requires the use of a feature function $\boldsymbol{\phi}$ to aggregate the collected time series Z^m_τ into a vector of features X^m_τ. Second, while modeling a maintenance problem using a POMDP with discrete observations is a common practice in the literature, a major difficulty of our problem lies in the choice of $\hat{f}^m$ to make the observations interpretable and discrete. Indeed, since the policy of our weakly coupled POMDP uses only the information in $O^m_t = \big(\hat{f}^m \circ \phi^m\big)\big((Z^m_\tau)_{\tau \leqslant t}\big)$, this policy can be relevant only if $\hat{f}^m$ is chosen in such a way that O^m_t provides relevant information on how the system evolves. But since the partition of $\mathbb{R}^{d^m}$ into n^m_O subspaces realized by $\hat{f}^m$ does not correspond to a ground-truth, there is no data from which it can be learned in a supervised learning way. Indeed, our dataset is not labeled. One way to learn $\hat{f}^m$ would therefore be to use unsupervised learning algorithms, but doing so, we have no way to indicate to the unsupervised learning algorithm that $\hat{f}^m$ should partition $\mathbb{R}^{d^m}$ in such a way that the different clusters are informative on the dynamic of the system. Therefore, we use the left-right HMM's predictions that assign an informative label to each feature vector X^m_τ. Learning a decision tree based on this labeling then make our approach combines interpretability and clustering.

An commonly used metric in predictive maintenance is the *Remaining Useful Life* (RUL), i.e., the time left before the next failure. We can evaluate the efficiency of our statistical model by measuring the accuracy of the predictions of the RUL within a validation methodology, which is commonly used in machine learning. However, our statistical approach is not a contribution to machine learning. What matters is the efficiency of the maintenance policy in terms of resulting saving costs and computation time. Nevertheless, as mentioned in the introduction of this chapter, in addition to the Gaussian HMM predictions building automatically a decision tree which gives an indicative label for each equipment corresponding to its degradation state is a useful contribution for Air France.

10.4 Numerical results

At Air France, the maintenance decision leverages a manually designed decision tree, that takes a continuous observation in input and returns a binary output $\{0, 1\}$. A failure is diagnosed on an equipment when the decision tree returns 1 and then a maintenance of the equipment is suggested. We would like to compare our approach against this current practice. However, we cannot evaluate it on real airline data for two reasons. First, we do not have an airplane's equipment simulator. Second, the historical dataset is censored, i.e., many equipments have been maintained before failing [132]. Since we do not have an airplane's component simulator, we cannot evaluate the performance of our policy on real data. We evaluate our methodology on a simulator of a deteriorating system with multiple components. Therefore, we construct a simulator of a deteriorating system based on the predictive maintenance literature and we reproduce the current practice on such a system. Then, we present the benefits of using our policy over the current practice on such system.

We also give several results on the real dataset of Air France. We compare the past decisions made by the current approach at Air France and what our policy would have done on the historical dataset. Even if the results have to be analyzed cautiously, one can observe a significant improvement over the current practice.

We use the library `scikitlearn` [118] for all machine learning algorithms. All linear programs have been implemented in `Julia` with package `JuMP` [41] and solved using `Gurobi 9.0` [52]. Experiments have been run on a server with 192GB of RAM and 32 cores at 3.30GHz. The code used to perform the numerical results on simulated data can be found at the following link `https://github.com/Victor2175/maintenance_system`.

10.4.1 Evaluating the policy using a simulator

System's description

We want to simulate a mechanical system composed of M deteriorating components. Several cracks are present in each component of the system. The deterioration of the component corresponds to the propagation of the cracks. Denote by $Z_\tau^m \in \mathbb{R}^{d^m}$ the noisy observation of the crack depth in component m at time τ. We suppose that for each component m, the dimension d^m has a moderate value (typically $3 \leqslant d^m \leqslant 5$) such that it does not require to perform feature extraction or feature selection. It means that our feature vector is equal to the sensor data at any time, i.e., $X_t^m := Z_{\tau_t}^m$. We assume that the crack depth in each component evolves independently. A complete description of how we simulate the crack depth propagations in the M components of a system is available in Appendix C.

Evaluation of a policy on the simulated system. We simulate the system over a time-period $\mathcal{T}$ with T_{sim} scheduled maintenance slots. We assume that the T_{sim} maintenance slots are periodically scheduled with interval time h, i.e., $\tau_t = t \times h$ is the time of maintenance slot t and $\mathcal{T} = T_{\mathrm{sim}} \times h$. Each component starts at time $\tau = 0$. At each maintenance slot $t \in [T_{\mathrm{sim}}]$, the decision maker receives the observation $Z^m_{\tau_t} = X_t \in \mathbb{R}^{d^m}$ on each component $m \in [M]$. According to a policy, if the decision maker maintains component m, then he pays the maintenance cost c^m_{m}. Otherwise, we let the system evolve until the next maintenance slot. If a failure happens between two maintenance slots, we suppose that the failure is observed. Then, the decision maker pays a failure cost c^m_{f}. We assume that when a component fails, the decision maker pays a failure cost and the component has been maintained.

Links with the Air France's maintenance problem. The simulated system aims at reproducing the airplane's maintenance problem. Indeed, the monitored equipments evolve independently. Between two maintenance slots the airplane is used to operate flights as shown in Figure 1.1. At each flight τ, the airline has access to some sensor data corresponding to $Z^m_\tau \in \mathbb{R}^{k^m}$ for each equipment $m \in [M]$. These continuous observations are in practice very noisy. We reproduce it in our simulator by adding a large noise to the crack depth measurements; see Appendix C. When the airplane goes into maintenance slot t, the airline decides to maintain at most K equipments. When a failure happens on an equipment between two maintenance slots, the equipment has to be maintained. Otherwise, the airplane is not able to take off again, which is what we mentioned in the last paragraph.

The system we simulate is a mechanical system. In the literature, the predictive maintenance is much more used on mechanical systems than electrical systems. Most of the equipments tracked by the airline are mechanical. It seems that there is no apparent reason that prevents the methodology of working in practice on electrical components of airplane. However, it requires a deeper work on the feature extraction step.

Reproducing the airline's maintenance policy. The airline's practice in industry is based on a decision tree with a binary output $\{0,1\}$. A component is maintained when the output of the decision tree is 1. We reproduce here such a policy for our system. Let g^m be a decision tree that takes as input a continuous observation x^m in $\mathbb{R}^{d^m}$ for all component m in $[M]$, and returns a binary output in $\{0,1\}$. The decision maker computes the vector $(g^m(x^m))_{m\in[M]}$. We maintain at most K components satisfying $g^m(x^m) = 1$. Thus, we introduce the set of components that should be maintained M^r,

$$M^r = \{m \in [M] \colon g^m(x^m) = 1\}.$$

If $|M^r| < K$, then we maintain all component m in M^r. Otherwise, we select the component m in M^r with the K highest failure cost c^m_{f}. Algorithm (6) describes how we reproduce the airline's current practice.

Algorithm 6 Airline's maintenance policy at a given time t

```
1: Input Decision trees $g^m$, observations $x^m$ for all $m$ in $[M]$.
2: Compute vector $\big(g^m(x^m)\big)_{m\in[M]}$.
3: Compute $M_r = \{m \in [M],\ \text{s.t. } g^m(x^m) = 1\}$.
4: if $|M^r| \geqslant K$ then
5:     Sort the components $m_1, \dots, m_{|M^r|}$ such that $c_{\mathrm{f}}^{m_1} \geqslant \dots \geqslant c_{\mathrm{f}}^{m_{|M^r|}}$.
6:     Maintain components $m_1, \dots, m_K$.
7: else
8:     Maintain all components in $M^r$.
9: end if
```

In Appendix C, we explain how we reproduce the airline's binary decision trees g^m for all m in $[M]$, using our simulator.

Numerical results. We apply our approach on the simulated system and we evaluate our policy against the current practice corresponding to Algorithm 6. We consider:

- our implicit policy (5.14) embed in the rolling horizon heuristic detailed in Algorithm 3 (Alg. 3) for different rolling horizons $T_{\mathrm{r}} = 1, 2, 5$
- the current practice in industry (Alg. 6)

Note that when $T_{\mathrm{r}} = 1$, Algorithm 3 becomes a greedy heuristic because the decision maker takes the action minimizing the expected costs over one time step. We evaluate each policy on different number of components $M \in \{3, 5, 10, 15, 20\}$. For each value of M, we set the maintenance capacity $K := \lfloor \frac{M+1}{3} \rfloor$. This choice is arbitrary but it enables to keep a fixed proportion regarding to the number of components M. We set the interval maintenance time between the maintenance slots $h = 30\Delta t$, where $\Delta t = 1$ is the discretization time step of our simulator. We also set the number of scheduled maintenance slots $T_{\mathrm{sim}} = 200$. For each value of M, we randomly draw 10 instances as described in Appendix C, where an instance corresponds to a set of parameters that fully describe a system. For each instance, we evaluate a policy 100 times, each time over the 200 time steps. In total, for each value of M, a policy is evaluated 1000 times. For each policy evaluation, we count the total cost and the number of failures at the end of the period. In addition, we calculate the mean time to take a decision during the evaluation, i.e., the mean computation time over the 200 time steps. As mentioned in Remark 16, the number of states $|\mathcal{X}_S^m|$ of the learned Gaussian HMMs is carefully chosen using the Bayes Information Criterion. For every instances, we obtain $3 \leqslant |\mathcal{X}_S^m| \leqslant 10$ for each component m.

Table 10.1 summarizes the results obtained. The first column indicates the number of components M. The second column indicates the policies used. Finally, the last three columns provide the policy computation time (Time), the value of objective function (Obj.) expressed as the percentage of cost saving over the airline's policy, and the number of failures (Fail.). All these quantities are averaged over the 1000 policy evaluations.

The results in Table 10.1 show that the maintenance policy of Algorithm 3 outperforms the airline's maintenance policy in terms of costs or failures, which is what we expected. In addition, we observe that using Algorithm 3 with $T_{\mathrm{r}} = 1$ already strongly outperforms the Air France's

M	K	Policy	Time (s)	Obj (%)	Fail.
3	1	Airline	-	-	10.0
		Alg. 3, $T_r = 1$	0.001	54.2	2.3
		Alg. 3, $T_r = 2$	0.010	54.8	1.6
		Alg. 3, $T_r = 5$	0.100	**57.9**	**1.0**
5	2	Airline	-	-	19.6
		Alg. 3, $T_r = 1$	0.003	55.7	4.0
		Alg. 3, $T_r = 2$	0.020	52.8	2.5
		Alg. 3, $T_r = 5$	0.290	**57.3**	**1.6**
10	3	Airline	-	-	35.2
		Alg. 3, $T_r = 1$	0.004	**53.5**	7.9
		Alg. 3, $T_r = 2$	0.030	50.6	5.4
		Alg. 3, $T_r = 5$	1.200	53.3	**4.3**
15	5	Airline	-	-	49.7
		Alg. 3, $T_r = 1$	0.008	**49.2**	12.6
		Alg. 3, $T_r = 2$	0.050	42.8	8.7
		Alg. 3, $T_r = 5$	2.000	47.5	**6.1**
20	7	Airline	-	-	69.4
		Alg. 3, $T_r = 1$	0.006	**55.1**	14.4
		Alg. 3, $T_r = 2$	0.040	47.2	9.3
		Alg. 3, $T_r = 5$	4.700	49.7	**5.2**

Table 10.1 – Numerical results on the simulated system averaged over the 1000 policy evaluations. The figures in bold indicate the best performances.

maintenance policy, and using a larger horizon gives almost the same total costs as Algorithm 3 with $T_r = 5$. Unfortunately, we are not able to explain precisely this phenomenon. But we try to give some explanations.

This phenomenon can be due to the fact that the weakly coupled POMDP parameters can be not well estimated for some instances. Indeed, the Gaussian HMM parameters are obtained by running the Baum-Welch algorithm, which usually reaches an local optimum of the likelihood instead of an global optimum. Depending on the initial conditions of the algorithm, this local optimum may lead to bad predictions. When we use Algorithm 3, the larger the rolling horizon T_r, the worse the predictions. Informally, it means that “if the predictions are not good for the next time step, then they will be worse over 2 or more time steps.” However, one can observe that the number of failures is significantly lower when we use a larger rolling horizon T_r, which means that the resulting maintenance policy is more preventive than with $T_r = 1$. These results emphasize that our maintenance policy strongly depend of the quality of the estimation of the HMM parameters. When the number of components grows, if more than one component has HMM parameters giving poor predictions, the maintenance policy can be worse.

10.4.2 Evaluating the policy on Air France real data

We present here some numerical results on data of Air France’s maintenance problem. Since we cannot simulate the equipment’s evolution over time, we propose an alternative method to evaluate our maintenance policy (5.14). We compare the past maintenance decisions against what our policy would have done. In our case, we have access to the sensor data of two equipments ($M = 2$) and the corresponding maintenance dates.

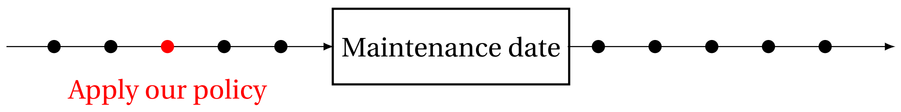


Figure 10.4 – Scheme of the evaluation of our maintenance policy. The black points indicate the flight legs. The red point corresponds to a flight leg.

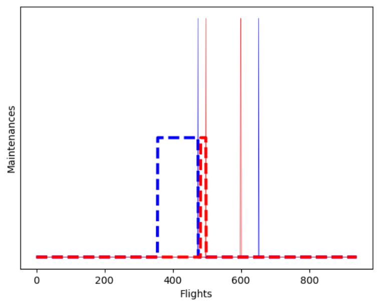
Handling the dataset, which contains all the sensor data, requires to use a Big Data processing engine. We choose to use `Spark` [167].

Evaluation of the maintenance policy. We compare what our maintenance policy would have done on the historical dataset against the past maintenance decisions of Air France. The available information are the maintenance dates of each equipment. When an equipment has been maintained, we know if the equipment has failed or not. However, as mentioned before, the data do not contain the exact failure dates.

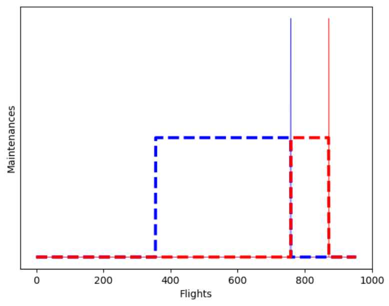
At each flight τ, we compute what our policy would have suggested on $(Z_\tau^m)_{m\in[M]}$. We compare the maintenance dates and the first flight when our policy would have suggested a maintenance. Figure 10.4 shows the scheme of the evaluation of our policy on the dataset. If the maintenance is a consequence of a failure, then we expect that the first maintenance suggestion would have appeared before the maintenance slot.

Numerical results on the AirFrance's dataset. By applying the policy evaluation scheme 10.4, we compare the maintenance dates and the dates where our maintenance policy would have suggested to maintain the equipment. Figure 10.5 illustrates our maintenance evaluation scheme on two different airplanes. After each flight, we apply our maintenance policy and the rising edges of the blue (resp. red) dashed line indicate when it suggests to maintain the equipment 1 (resp. 2). The sizes of the state space and the observation space we obtain by using the methodology of Section 10.3 are $|\mathcal{X}_S^m| = |\mathcal{X}_O^m| = 8$ for every m in $\{1,2\}$. Hence, the sizes of the state space and of the observation space of the full system are $|\mathcal{X}_S| = |\mathcal{X}_O| = 64$ and the size of the action space is $|\mathcal{X}_A| = 3$.

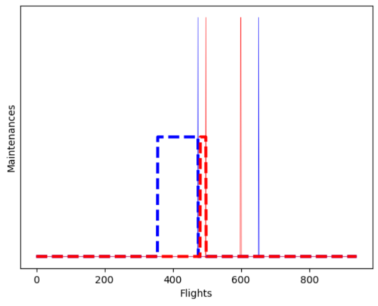
A maintenance date of an equipment (vertical plain line in Figure 10.5) does not necessary indicate that the equipment failed. Indeed, in most of the cases the maintenances are preventive and the equipment has not failed when it it maintained. When a failure has been diagnosed in maintenance, it means that Air France's approach did not predict it in advance. To evaluate our maintenance policy, we count the percentage of failures such that our maintenance policy would have suggested to maintain the equipment before the corresponding maintenance dates, among those that have been diagnosed in maintenance. We denote by $\mathrm{f}_{T_\mathrm{r}}^m$ such a percentage for equipment m and obtained by using Algorithm 3 with rolling horizon T_r. On the whole fleet, we count 10 diagnosed failures for each equipment. Table 10.2 reports all the values of $\mathrm{f}_{T_\mathrm{r}}^m$ obtained on the whole fleet. The first column indicates the rolling horizon T_r we use in our policy. The second and third columns indicate respectively the values of $\mathrm{f}_{T_\mathrm{r}}^1$ and $\mathrm{f}_{T_\mathrm{r}}^2$. Finally, the last column indicates the averaged computation time (Comp. Time) to take a decision at each flight.



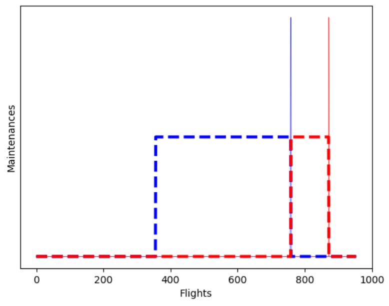
(a) Algorithm 3 with $T_{\mathrm{r}} = 2$.



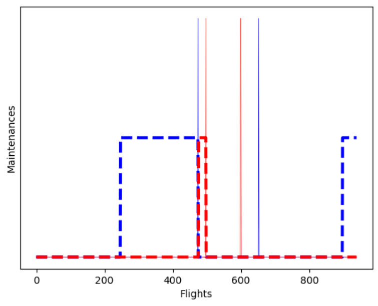
(b) Algorithm 3 with $T_{\mathrm{r}} = 2$.



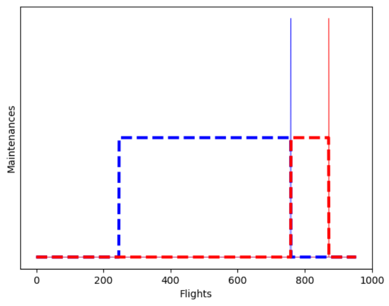
(c) Algorithm 3 with $T_{\mathrm{r}} = 5$.



(d) Algorithm 3 with $T_{\mathrm{r}} = 5$.



(e) Algorithm 3 with $T_{\mathrm{r}} = 10$.



(f) Algorithm 3 with $T_{\mathrm{r}} = 10$.

Figure 10.5 – An example of the application of our maintenance policy on two airplanes (one for each column) of Air France's dataset. The horizontal axis represents the flights. The blue (resp. red) plain vertical lines indicate the flights after which a maintenance has been performed on equipment 1 (resp. equipment 2). The blue (resp. red) dashed lines represent the maintenance suggestion of our maintenance policy for equipment 1 (resp. equipment 2). The rising edges of the dashed lines indicate when our maintenance policy would have suggested to maintain the equipment.

Rol. horizon	$f^1_{T_r}$ (%)	$f^2_{T_r}$ (%)	Comp. Time (s)
$T_r = 1$	00	10	0.006
$T_r = 2$	30	20	0.030
$T_r = 5$	30	20	0.620
$T_r = 10$	30	40	2.700

Table 10.2 – Numerical values of $f^m_{T_r}$ for the whole fleet.

Note that these results have to be considered carefully because we do not know when the failure happens exactly, which means that our maintenance policy could have suggested to maintain an equipment that has already failed. Table 10.2 shows that our policy would have avoided several failures and outperforms Air France's policy. Since the results obtained are at least as good as Air France's results, we can reasonably say that the learned decision tree that predicts the state of the equipment is efficient. It is also a contribution for Air France. In addition, one can observe that the computation time is always below several seconds, which is an advantage for Air France.

10.5 Bibliographical remarks

A *data-driven maintenance optimization* approach learns a statistical model from the available data, and, leverages it to derive an optimized maintenance decision when new data becomes available. The tools we used to build the statistical model, feature extraction, HMMs and decision trees, are known in the predictive maintenance literature. Kim et al. [72] proposed a similar data-driven methodology on the maintenance of heavy hauler truck used in mining industry and they showed that their approach enables to save 34% of the cost with respect to the current practice.

Since our approach uses some tools from different research areas, we divide the literature review in two sections. First, we review the current practice on the feature extraction, the HMMs and the decision trees in predictive maintenance. Second, we review the use of POMDP in maintenance. We add here bibliographical remarks about the three steps to build the statistical model described in this chapter: features extraction, Hidden Markov Model and decision tree.

Features extraction is a common technique in machine learning to reduce the dimension of data by aggregating the input data in a finite set of features. The resulting dimension is the number of features. For an overview of feature extractions and signal processing with maintenance sensor data, see for instance Tsui et al. [153, Sec 2.1], Jouin et al. [63, Sec 4.1.2] or Gouriveau and Zerhouni [51, Chapter 3]. In order to keep the most relevant features, Javed et al. [61] propose an approach based on the notion of feature predictability, i.e., the ability of a feature to be predicted by a state-of-the-art time series prediction algorithm. In particular they show that the more predictable the features, the better the failure predictions. We choose to stick to the predictive maintenance literature. We compute a moderate number of simple features for time series [153, Table 1] such that we obtain observations in $\mathbb{R}^d$, where d is between 10 and 50.

Hidden Markov Model (HMM) is one of the main tools to model a component's evolution with observations in $\mathbb{R}^d$. The component's degradation is modeled as a hidden Markov chain in a finite state space, and we assume that at each time the continuous observation in $\mathbb{R}^d$ depends on the current hidden state. An HMM is parameterized by its conditional probability distributions. This modeling has been widely used in maintenance industry [82, 142, 151, 160]. In particular, HMMs are appreciated in such a context due to their ability to predict the *Remaining Useful Life* (RUL) [161]. It is for this reason that HMMs lie in the broad class of *prognostic* methods [60]. Like Le et al. [82], we model the evolution of the continuous observations in $\mathbb{R}^d$ using a Gaussian HMM. However, an expert cannot interpret the RUL's predictions made by the HMM. Hence, it requires to make the HMM's observations interpretable by a maintenance expert using a decision tree.

Decision trees are widely used in the industry to detect failures. Given a vector of features in $\mathbb{R}^d$ in input, it returns a discrete label, which corresponds to a failure risk cluster of the system. As mentioned before, the decision trees are appreciated in the industry because they are interpretable [39, 47]. While in the airline industry the decision trees are hand designed by maintenance experts, machine learning algorithms allow to learn decision trees that can be interpreted as decision trees [134]. Decision tree learning approaches have been used in various maintenance applications [26, 131, 150]. In all of them, the decision trees are learned using a training dataset with labeled data, and algorithm C4.5 or CART [19, 53]. For each observation in the dataset, a label indicates the presence of a failure and the decision tree aims at predicting this label.

11 Conclusion

As a conclusion, we summarize the main contributions of this thesis, and outline some research directions suggested by our results. Our work was applied to the case of Air France, which was our partner during the thesis.

11.1 Main contributions

In this dissertation, we have developed algorithms for stochastic optimization problems including the Partially Observable Markov Decision Process problem and the maximum expected utility problem in influence diagrams. The first and second parts are mainly theoretical and introduce mathematical programming formulations to solve these stochastic optimization problems. The third part applies one of these algorithms to solve the airplane maintenance problem at Air France.

Part I introduces a generic predictive maintenance problem with capacity constraints. We consider a system with several components, each of them evolving independently over time. At each maintenance slot, based on an observation of each component the decision maker chooses which components to maintain. The actions taken on each component are linked by the capacity constraints. This choice is modeled using a memoryless policy that maps a vector of observations to an action. The objective is to choose a memoryless policy minimizing the expected failure costs and maintenance costs over a finite horizon. This problem lies in the broad class of weakly coupled dynamic programs [2, 55]. An additional feature of our problem comes from the fact that the components are partially observable. While modeling the deterioration of a component using a POMDP is standard in the literature [1, 42, 91], we introduce the weakly coupled POMDP that models a system with several components, each of them evolving independently as a POMDP. We emphasize the modeling power of the weakly coupled POMDP on several practical problems.

To provide a good policy for the weakly coupled POMDP, we made several contributions for the POMDP problem with memoryless policies, which is NP-hard. First, we proposed an mixed-integer linear program that gives an optimal memoryless policy for POMDP. The variables of this MILP are the marginal probability distributions of the random variables, and the constraints are the ones satisfied by a joint probability distribution over the random variables. Second, based on a probabilistic interpretation of the dependences between the random variables, we introduced valid inequalities for our MILP that improve its resolution. The numerical results in Chapter 4 show that these valid inequalities tighten significantly the linear relaxation. Third, we show how to relate the value of our MILP to the value of the usual POMDP where the actions are taken based on the history of actions and observations. We evaluate the performances of memoryless policies on several POMDP instances from the literature over finite horizon. The numerical results show that the memoryless policies perform well on a large part of the instances, including those corresponding to maintenance problems.

Like weakly coupled dynamic programs, the weakly coupled POMDPs suffer from the curse of dimensionality because the sizes of the state space and the observation space of the full system are exponential in the number of components. Even encoding a memoryless policy is intractable. To overcome this issue, we introduce an MILP containing a polynomial number of variables and constraints, which breaks the curse of dimensionality, and we give theoretical guarantees on its optimal value by playing with different "probabilistic" approximations. In particular, like [2] for the weakly coupled dynamic programs (or weakly coupled MDPs) we use the Lagrangian relaxation approach to derive a tractable upper bound. In Chapter 5, we illustrate the quality of our approximation on instances of multi-armed bandit problems. Leveraging this MILP, we define a feasible policy of the weakly coupled POMDP that provides satisfying results for the predictive maintenance problem with capacity constraints in Chapter 10.

Part II focuses on stochastic optimization problems including MDPs and POMDPs where the uncertainty satisfies some structure of influence diagram. Chapter 6 describes our main results for influence diagrams, which generalize the integer programs of Part I on POMDPs to influence diagrams. We introduce linear programming and MILP approaches for the maximum expected utility problem in influence diagrams. The variables of the programs correspond to the collection of vector of moments of the distribution on subsets of the variables that are associated to vertices of a new kind of junction tree, which we call an rooted junction tree. We have also proposed algorithms to build rooted junction trees tailored to our linear and integer programs. In Chapter 8 we proposed an MILP approach to solve the maximum expected utility problem on influence diagrams together with valid cuts. Again, these valid cuts are based on a probabilistic interpretation and the numerical experiments in Chapter 8 show that the bound obtained with this linear relaxation is indeed better in practice than without these inequalities.

In Chapter 9, we study soluble influence diagrams [81], which are influence diagrams whose maximum expected utility problem is easy, in the sense that it can be solved by the algorithm "single policy update"(SPU). We show that for soluble influence diagrams the maximum expected utility problem can also be solved exactly via our linear programs. Furthermore, we characterized soluble influence diagrams as the influence diagrams for which there exists a junction tree such that the set of possible vectors of moments on the vertices of the tree is convex for any parameterization of the influence diagram. The bound provided by the linear relaxation is better than the bound that could be obtained using SPU on a soluble relaxation.

Part III focuses on the airplane maintenance problem at Air France. While the predictive maintenance problem with capacity constraints formalized in Part I considers discrete observations, the decision maker of the airplane maintenance problem has access to sensor data recorded during flights. We describe a statistical methodology to cast this airplane maintenance problem as a weakly coupled POMDP problem. In particular, based on a dataset of sensor data we explain how we estimate the POMDP parameters of each equipment of the airplane. One of the main advantages of our approach is that all its steps are interpretable by maintenance experts, which means that the observations of the POMDPs are the output of a decision tree. Since we do not have access to a simulator of the sensor data, we evaluate our weakly coupled POMDP policy using a simulator of a system with several deteriorating components, which is built from the literature. In addition, we compare what our maintenance policy would have suggested in the historical dataset against Air France's decisions. In both cases, the numerical experiments show that our weakly coupled POMDP policy outperforms the one used by Air France.

Along this thesis, we developed the ideas on influence diagrams and detailed them in the following published paper.

Axel Parmentier, Victor Cohen, Vincent Leclère, Guillaume Obozinski and Joseph Salmon. Integer Programming on the Junction Tree Polytope for Influence Diagrams. *INFORMS Journal on Optimization*, 2(3):209–228, 2020

All the mathematical programs with the value function variables came later, and Part II extends the content of this paper. Some of the results on POMDPs in Chapter 4 have been partially detailed in the following preprint.

Victor Cohen and Axel Parmentier. Linear Programming for Decision Processes with Partial Information. arXiv preprint arXiv:1811.08880, 2018

Articles based on the results of Part I and Part III are currently being finalized.

11.2 Research directions

We now highlight some research directions raised by our work on POMDPs and the influence diagrams.

In Chapter 5, we introduced different integer programs with theoretical guarantees for the weakly coupled POMDP. In particular, most of these approximations are based on a surrogate modeling of the linking capacity constraints. However, as we observe in the numerical experiments of multi-armed bandit instances, the integer programs become difficult to solve when the size of the state spaces and observation spaces increase. We could investigate a Branch-and-price algorithm based on the Dantzig-Wolfe decomposition of the integer programs. Leveraging the column generation approach, this algorithm would give an optimal solution of MILP (5.1) and could be efficient on large scale instances.

We are currently working on an approximate integer formulation for ($\mathrm{P}^{\mathrm{wc}}_{\mathrm{ml}}$) using the value function variables. The resulting formulation could be a generalization of the ones of Adelman and Mersereau [2] to weakly coupled POMDPs.

In Part II, we proposed mathematical programming approaches for solving the maximum expected utility problem in influence diagrams. Two elements limit the scale of the problems that can be dealt with using our approach. First, we use exact inference, which limits the applicability to models with small treewidth. Second, rooted junction trees may contain clusters larger than those of standard junction trees. A possible way to overcome these limitations in future works would be to develop mathematical programming heuristics for influence diagrams that use variational inference instead of exact inference. Several works in graphical models that propose linear programming approaches for variational inference could be investigated [148, 149]. We are working on a method to use inference techniques such as Bethe entropy approximation [156, Chapter 4] for influence diagrams.

We are also working on a distributionally robust version of the maximum expected utility problem in influence diagrams. In Part II, we assumed that the conditional probability distributions of the chance vertices $\mathfrak{p} = (p_{v|\mathrm{pa}(v)})_{v\in V^{\mathrm{s}}}$ are known by the decision maker. However, in practice these parameters are not always available. Indeed, the decision maker has usually access to N realizations of the random variables of the influence diagrams $(X_V^i)_{i=1,\dots,N}$. Based on this dataset, the usual approach we used for the airplane maintenance problem at Air France is to compute approximate parameters $\hat{\mathfrak{p}}$ and then to find a feasible strategy $\boldsymbol{\delta}$ of the maximum expected utility problem. However, as we observe in the numerical experiments of Section 10, such an approach may give misleading strategies with poor results. To overcome this issue, we suggest to study the following distributionally robust optimization problem

$$\max_{\boldsymbol{\delta}} \min_{\mathfrak{p}\in\mathcal{N}(\hat{\mathfrak{p}})} \mathbb{E}_{\mathfrak{p},\boldsymbol{\delta}}\Big[\sum_{v\in V^{\mathrm{r}}} r_v(X_v)\Big],$$

where the expectation is taken according to the probability distribution $\mathbb{P}_{\mathfrak{p},\boldsymbol{\delta}}$ and $\mathcal{N}(\mathfrak{p})$ defines a neighborhood around the estimate $\hat{\mathfrak{p}}$. This type of distributionally robust optimization problems have received interest in the last decades [36, 45, 126] for problem with exogenous noises. In influence diagram problems, the decisions and the "nature" are not independent. Recently, several works have proposed theoretical studies for robust MDPs [163] and robust POMDPs [163]. We wish to extend our mathematical programming approaches for this robust version of the maximum expected utility problem with theoretical guarantees.

Appendix Part

A Examples where $z_{\mathrm{IP}} < v^*_{\mathrm{ml}}$ or $z_{\mathrm{IP}} > v^*_{\mathrm{ml}}$

In this section, we describe two instances showing respectively that MILP (5.1) is neither an upper bound nor a lower bound. We denote by z_{IP} the optimal value of MILP (5.1).

A.1 The inequality $z_{\mathrm{IP}} \leqslant v^*_{\mathrm{ml}}$ does not hold in general

Consider a weakly coupled POMDP with $M = 2$, $K = 1$, $\mathcal{X}_S^1 = \mathcal{X}_S^2 = \{1,2,3\}$, and $\mathcal{X}_O^1 = \mathcal{X}_O^2 = \{1,2\}$. We set the following initial probability data,

$$p^1(\cdot) = \begin{bmatrix} 0.0286 & 0.4429 & 0.5284 \end{bmatrix}, \qquad p^2(\cdot) = \begin{bmatrix} 0.5328 & 0.2202 & 0.2469 \end{bmatrix},$$

the following transition probability data,

$$p^1(\cdot|\cdot,0) = \begin{bmatrix} 0.3149 & 0.2598 & 0.4253 \\ 0.2542 & 0.5195 & 0.2263 \\ 0.2016 & 0.7551 & 0.0433 \end{bmatrix}, \qquad p^2(\cdot|\cdot,0) = \begin{bmatrix} 0.6833 & 0.1797 & 0.1371 \\ 0.0398 & 0.9207 & 0.0394 \\ 0.1422 & 0.2202 & 0.6376 \end{bmatrix},$$

$$p^1(\cdot|\cdot,0) = \begin{bmatrix} 0.3849 & 0.2891 & 0.3260 \\ 0.4462 & 0.1346 & 0.4192 \\ 0.0418 & 0.5297 & 0.4285 \end{bmatrix}, \qquad p^2(\cdot|\cdot,1) = \begin{bmatrix} 0.4665 & 0.0956 & 0.4379 \\ 0.4510 & 0.5168 & 0.0322 \\ 0.5864 & 0.2903 & 0.1234 \end{bmatrix},$$

the following emission probability data,

$$p^1(\cdot|\cdot) = \begin{bmatrix} 0.6823 & 0.3177 \\ 0.0806 & 0.9194 \\ 0.5018 & 0.4982 \end{bmatrix}, \qquad p^2(\cdot|\cdot) = \begin{bmatrix} 0.4389 & 0.5611 \\ 0.6657 & 0.3343 \\ 0.1207 & 0.8793 \end{bmatrix},$$

and the following reward data

$$r^1(\cdot,0,\cdot) = \begin{bmatrix} 3.3101 & 7.8198 & 6.9773 \\ 2.0722 & 2.6782 & 3.5715 \\ 8.4428 & 2.6010 & 3.2765 \end{bmatrix}, \qquad r^2(\cdot,0,\cdot) = \begin{bmatrix} 2.9600 & 8.1503 & 4.5911 \\ 2.2638 & 6.0290 & 2.5511 \\ 8.0789 & 7.9927 & 5.0259 \end{bmatrix},$$

$$r^1(\cdot,1,\cdot) = \begin{bmatrix} 1.9315 & 9.3614 & 2.8927 \\ 4.8769 & 5.3131 & 7.3626 \\ 3.7944 & 4.5557 & 8.6462 \end{bmatrix}, \qquad r^2(\cdot,1,\cdot) = \begin{bmatrix} 6.2647 & 6.6832 & 1.1263 \\ 9.9182 & 9.0278 & 5.9492 \\ 9.8333 & 0.4466 & 4.3798 \end{bmatrix}.$$

Solving $\text{P}^{\text{wc}}_{\text{ml}}$ with $T = 4$ using MILP (4.7) on $\mathcal{X}_S$, $\mathcal{X}_O$ and $\mathcal{X}_A$, we obtain an optimal value of $v^*_{\text{ml}} = 44.7122$, while the optimal value of our MILP (5.1) is $z_{\text{IP}} = 44.2834$. Hence, we obtain $z_{\text{IP}} < v^*_{\text{ml}}$. Therefore, $v^*_{\text{ml}} \leqslant z_{\text{IP}}$ does not hold in general.

A.2 The inequality $z_{\text{IP}} \geqslant v^*_{\text{ml}}$ does not hold in general

Consider a weakly coupled POMDP with $M = 2$, $K = 1$, $\mathcal{X}^1_S = \mathcal{X}^2_S = \{1,2,3\}$, and $\mathcal{X}^1_O = \mathcal{X}^2_O = \{1,2\}$. We set the following initial probability data,

$$p^1(\cdot) = \begin{bmatrix} 0.4311 & 0.5255 & 0.0434 \end{bmatrix}, \qquad p^2(\cdot) = \begin{bmatrix} 0.4835 & 0.1745 & 0.3421 \end{bmatrix},$$

the following transition probability data,

$$p^1(\cdot|\cdot,0) = \begin{bmatrix} 0.1517 & 0.3481 & 0.5002 \\ 0.1639 & 0.0922 & 0.7439 \\ 0.3395 & 0.2385 & 0.4220 \end{bmatrix}, \qquad p^2(\cdot|\cdot,0) = \begin{bmatrix} 0.3435 & 0.3291 & 0.3274 \\ 0.5964 & 0.1653 & 0.2383 \\ 0.3968 & 0.2626 & 0.3406 \end{bmatrix},$$

$$p^1(\cdot|\cdot,1) = \begin{bmatrix} 0.3467 & 0.2733 & 0.3800 \\ 0.5027 & 0.3548 & 0.1425 \\ 0.2530 & 0.5466 & 0.2003 \end{bmatrix}, \qquad p^2(\cdot|\cdot,1) = \begin{bmatrix} 0.3160 & 0.4210 & 0.2630 \\ 0.3583 & 0.3882 & 0.2535 \\ 0.3611 & 0.4308 & 0.2081 \end{bmatrix},$$

the following emission probability data,

$$p^1(\cdot|\cdot) = \begin{bmatrix} 0.2052 & 0.7948 \\ 0.8296 & 0.1704 \\ 0.5330 & 0.4670 \end{bmatrix}, \qquad p^2(\cdot|\cdot) = \begin{bmatrix} 0.6273 & 0.3727 \\ 0.0392 & 0.9608 \\ 0.4024 & 0.5976 \end{bmatrix},$$

and the following reward data

$$r^1(\cdot,0,\cdot) = \begin{bmatrix} 7.0075 & 6.2135 & 8.4122 \\ 9.7198 & 9.5152 & 2.6182 \\ 1.8522 & 7.4390 & 4.9132 \end{bmatrix}, \qquad r^2(\cdot,0,\cdot) = \begin{bmatrix} 8.7418 & 2.6682 & 2.5227 \\ 8.7673 & 6.1198 & 6.4814 \\ 6.4971 & 3.8810 & 0.3476 \end{bmatrix},$$

$$r^1(\cdot,1,\cdot) = \begin{bmatrix} 2.8154 & 7.0215 & 1.6752 \\ 7.8149 & 0.7849 & 4.3722 \\ 5.9378 & 9.1273 & 1.1657 \end{bmatrix}, \qquad r^2(\cdot,1,\cdot) = \begin{bmatrix} 7.4528 & 8.5013 & 9.1925 \\ 4.3003 & 2.0946 & 4.2973 \\ 4.2865 & 0.8470 & 9.5848 \end{bmatrix}.$$

Solving $\text{P}^{\text{wc}}_{\text{ml}}$ with $T = 4$ using MILP (4.7) on $\mathcal{X}_S$, $\mathcal{X}_O$ and $\mathcal{X}_A$, we obtain an optimal value of $v^*_{\text{ml}} = 47.3693$, while the optimal value of our MILP (5.1) is $z_{\text{IP}} = 47.7356$. Hence, we obtain $v^*_{\text{ml}} < z_{\text{IP}}$. Therefore, $v^*_{\text{ml}} \geqslant z_{\text{IP}}$ does not hold in general.

B Algorithm to build a small RJT

In this appendix we present an algorithm to build a RJT without considering a topological ordering on the initial graph $G = (V, A)$.

The only difference between Algorithms 4 and 7 is that the for loop along a reverse topological ordering of Algorithm 4 is replaced in Algorithm 7 by a breadth first search that computes online this reverse topological ordering. Hence, if we denote $\preccurlyeq$ this ordering, Algorithm 7 builds the same RJT as the one we obtain when we use Algorithm 4 with $\preccurlyeq$ in input. Therefore, the RJT built by Algorithm 7 satisfies 7.6, and is such that the implications in (7.7) are equivalence.

Furthermore, Steps 5 and 6 enable to ensure that, when there is no path between a vertex $u \in V^{\mathrm{a}}$ and a vertex $v \in V^{\mathrm{s}}$, then u is placed before v in the reverse topological ordering computed by the breadth first search. Therefore, $\preccurlyeq$ is a topological ordering on the graph G'' used as Step 9 of Algorithm 5. Hence, if G is soluble, Algorithm 7 builds a RJT such that $G^{\perp\!\!\perp} = G$.

Remark that on non-soluble IDs, Steps 5 and 6 are a heuristic aimed at minimizing the size of $C_v^{\perp\!\!\perp}$ for each v in V^{s}. Such a heuristic is not relevant if valid cuts (6.7) are not used. In that case, an alternative approach could be to add as few variable as possible to C_v for v in V^{a} to improve the quality of the soluble relaxation $\overline{G}$. This could be done by putting vertices u in V^{s} unrelated to $v \in V^{\mathrm{a}}$ after in this topological order, i.e., by replacing V^{a} by V^{s} in Steps 5 and 6.

Algorithm 7 Build a RJT

```
1: **Input** $G = (V, E)$
2: **Initialize** $\mathcal{C} = \emptyset$ and $\mathcal{A}' = \emptyset$
3: $L = \{v \text{ s.t } \mathrm{ch}_G(v) = \emptyset\}$
4: **while** $L \neq \emptyset$ **do**
5:     **if** $V^{\mathrm{a}} \cap L \neq \emptyset$ **then**
6:         Pick $v \in V^{\mathrm{a}} \cap L$
7:     **else**
8:         Pick $v \in L$
9:     **end if**
10:    $C_v \leftarrow \mathrm{fa}(v)$
11:    **for** $C_x \in \mathcal{C} : v \in C_x$ **do**
12:        $C_v \leftarrow C_v \cup (C_x \setminus \{x\})$
13:        Remove $C_x$ from $\mathcal{C}$
14:        Add $\{v, x\}$ in $\mathcal{A}'$
15:    **end for**
16:    Add $C_v$ to $\mathcal{C}$
17:    Remove $v$ from $G$ and recompute $L$
18: **end while**
19: $\mathcal{A} \leftarrow \{(C_u, C_v) \mid (u, v) \in \mathcal{A}'\}$
20: **Return** $\mathcal{G} = ((C_v)_{v \in V}, \mathcal{A})$
```

C Deteriorating system's simulator and decision trees

In Section C.1 of this appendix we describe the system we simulate to evaluate the performances of our maintenance policy in Chapter 10. In Section C.2, we explain how we reproduce the Air France's practice for the system we simulate.

C.1 Simulator's description

Fatigue Crack Growth (FCG) models the appearance and propagation of a crack within a bearing. It is a widely used model in the predictive maintenance literature [31, 82, 109, 161] to simulate a deteriorating system. Each coordinate $k \in [d]$ of the crack depth y^k evolves according to the following Paris-Erdogan differential equation [115]

$$\frac{\mathrm{d}y^k}{\mathrm{d}t} = C(\beta e^{\gamma}\sqrt{y^k})^n \tag{C.1}$$

where C and n are parameters depending on the material property, β is the base stress level of the component and γ determines the extra stress level of the component. This latter parameter depends on the environment state. The greater γ is, the faster the crack propagates.

Simulation We simulate the cracks in a component with FCG model where the extra stress level is a random variable. Let Γ_i be such random variable at time i. The system's extra stress level transits randomly from $\Gamma_i = \gamma$ to $\Gamma_{i+1} = \gamma'$ with probability $p_\Gamma(\gamma'|\gamma)$ for all $\gamma, \gamma' \in \mathcal{X}_\Gamma$. We assume that the crack propagation rate stagnates or increases:

$$p_\Gamma(\gamma'|\gamma) = 0 \text{ if } \gamma' < \gamma \tag{C.2}$$

We combine this model with the discretization scheme detailed in Le et al. [82] of Equation (C.1). Let γ_i and $(y_i^k)_{1\leqslant k\leqslant d}$ be respectively the extra stress level and the crack length at time i:

$$\begin{aligned} &\Gamma_{i+1} \sim p_\Gamma(.|\gamma_i) \\ &y_{i+1}^k = y_i^k + e^{W_{i+1}} C\left(\beta e^{\Gamma_{i+1}}\sqrt{y_i^k}\right)^n \Delta t \end{aligned} \tag{C.3}$$

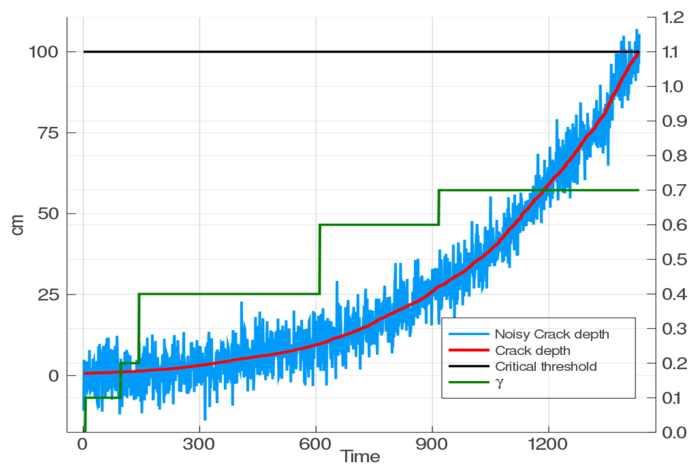


Figure C.1 – Example of the crack depth propagation in a component along one dimension. The simulation is obtained according to Scheme (C.3). Horizontal axis represents the time in days. The left and right vertical axis respectively represent the crack depth in cm and the extra-stress level range. The red and blue curves respectively correspond to the crack depth y_i and the noisy crack depth x_i. The green curve represents the evolution of the extra stress level γ_i over time. The black horizontal line represents the critical threshold y_c.

where $W_i \sim \mathcal{N}(0, \Sigma_w)$, $\Sigma_w \in \mathbb{R}^{d\times d}$, and Δt represents the discretization time step of the scheme. Note that the stochasticity of scheme (C.3) comes from W_i and Γ_i. We assume that the component fails when the crack length reaches a critical threshold y_c. Hence, a failure happens at i when there is at least one coordinate $k \in [d]$ such that $Y_i^k \geqslant y_c$.

Since we have noisy measurements X_i of the crack depth Y_i, we add a white noise:

$$X_i = Y_i + \xi$$

where $\xi \sim \mathcal{N}(0, \Sigma_\xi)$ with $\Sigma_\xi \in \mathbb{R}^{d\times d}$ is a measurement error. Figure C.1 shows an example of trajectory generated using the discretization Scheme (C.3) with parameters selected as detailed below.

Parameter settings of a simulation Given a number of component M, the maintenance capacity K and the interval time between two maintenance slots h, we describe how we set the parameters of a simulation. First we set the maintenance costs C_R^m and the failure costs C_F^m such that $C_F^m = 10 \times C_R^m$ for each component $m \in [M]$. Second, we set the parameters of Scheme (C.3): y_c, Δt, C, n, β, Σ_w, Σ_ξ, the transition probabilities p_Γ and the space $\mathcal{X}_\Gamma$. For each component, we choose the same critical threshold $y_c = 100\,\text{cm}$ and same discretization time step $\Delta t = 1$ as Le et al. [82]. Since each component $m \in [M]$ evolves independently, we set a combination of parameters C^m, n^m, β^m, Σ_w^m, Σ_ξ^m, $\mathcal{X}_\Gamma^m$ and p_Γ^m. We randomly draw scalar parameters in their respective range $C^m \in [0.001, 0.1]$, $n^m \in [1,2]$, $\beta^m \in [0.5, 1.5]$. Then, we set the parameters of probability distributions. Like Le et al. [82], we randomly draw a covariance matrix Σ_w^m such that $\left|\Sigma_w^m\right| = 1.7$. As mentioned above the data are in practice very noisy, hence we set a large Frobenius norm value for Σ_ξ^m. We randomly draw a covariance matrix Σ_ξ^m such that $\left|\Sigma_{\xi}^{m}\right| = 100$. Finally, we randomly draw the size of the finite space $|\mathcal{X}_{\Gamma}^{m}|$ in $\{5,\dots,15\}$ and the transition probabilities p_{Γ}^{m} satisfying (C.2). Then we split interval $[0,1.2]$ into $|\mathcal{X}_{\Gamma}^{m}|-1$ intervals and we set $\mathcal{X}_{\Gamma}^{m} = \left\{\frac{1.2k}{|\mathcal{X}_{\Gamma}^{m}|-1},\ k=0,\dots,|\mathcal{X}_{\Gamma}^{m}|-1\right\}$.

C.2 How to reproduce a binary decision tree of AirFrance's practice.

Now, we describe how we learn the binary decision trees g^m used in Algorithm 6, for any component $m \in [M]$. For convenience, we omit the index m in the rest of this section.

Labeling the data. Let $y_1,\dots,y_T \in \mathbb{R}^d$ be the crack depth evolution generated according to (C.3) until a failure. Therefore, $y_T^i \geqslant y_{critic}$ for at least one i in $[d]$. Let q^i be the 90-th percentile of the i-th coordinate of the crack depth. For each data point y_t, we affect a binary label $z_t \in \{0,1\}$ as follows

$$z_t = \prod_{i=1}^{d} \mathbb{1}_{y \geqslant q_i}(y_t^i) \tag{C.4}$$

Hence, z_t indicates the early stages of a failure.

Learning the decision tree. Given a sequence $y_1,\dots,y_T$ in $\mathbb{R}^d$ generated according to (C.3), we compute the corresponding sequence of labels $z_1,\dots,z_T$ in $\{0,1\}$ using (C.4). We generate a large number of sequences and we compute the corresponding sequence of labels. Then, we use the CART Algorithm [19] to learn a decision tree $g\colon \mathbb{R}^d \to \{0,1\}$ that takes in input the observation $x_t \in \mathbb{R}^d$ and in output the label $z_t \in \{0,1\}$.

Bibliography

[1] Abderrahmane Abbou and Viliam Makis. Group maintenance: A restless bandits approach. *Journal on Computing*, 31:719–731, 2019.

[2] Daniel Adelman and Adam J. Mersereau. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56:712–727, 2008.

[3] Suzan Alaswad and Yisha Xiang. A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering & System Safety*, 157:54 – 63, 2017.

[4] Alessandro Antonucci and Marco Zaffalon. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *International Journal of Approximate Reasoning*, 49:345 – 361, 2008.

[5] Raghav Aras and Alain Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37:329–396, 2010.

[6] Raghav Aras, Alain Dutech, and François Charpillet. Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. In *Proceedings of the Seventeenth International Conference on International Conference on Automated Planning and Scheduling*, pages 18–25, 2007.

[7] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Open problem: Approximate planning of POMDPs in the class of memoryless policies. In *Proceedings of Machine Learning Research*, volume 49, pages 1639–1642, 2016.

[8] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning of POMDPs using spectral methods. In *Proceedings of Machine Learning Research*, volume 49, pages 193–256, 2016.

[9] J. Andrew Bagnell, Sham M. Kakade, Jeff G. Schneider, and Andrew Y. Ng. Policy search by dynamic programming. In *Advances in Neural Information Processing Systems 16*, pages 831–838. 2004.

[10] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[11] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[12] Richard Bellman. Adaptive control processes—a guided tour. *Princeton University Press*, 8:315–316, 1961.

[13] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[14] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition, 2007.

[15] Dimitris Bertsimas and Velibor V Mišić. Decomposable Markov decision processes: A fluid optimization approach. *Operations Research*, 64:1537–1555, 2016.

[16] Dimitris Bertsimas and José Niño Mora. Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21:257–306, 1996.

[17] Dimitris Bertsimas and José Niño Mora. Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research*, 48:80–90, 2000.

[18] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59:65–98, 2017.

[19] Leo Breiman, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[20] David B. Brown, James E. Smith, and Peng Sun. Information relaxations and duality in stochastic dynamic programs. *Operations Research*, 58:785–801, 2010.

[21] Andrés Cano, José E. Cano, and Serafín Moral. Convex sets of probabilities propagation by simulated annealing. In *Proceedings of the Fifth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 4–8, 1994.

[22] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61, 1997.

[23] Anthony R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, 1998.

[24] Anthony R. Cassandra. A survey of POMDP applications. In *Working note of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes.*, pages 17–24, 1998.

[25] Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. Complexity of inference in graphical models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 70–78, 2008.

[26] Mike Chen, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *Proceedings of the First International Conference on Autonomic Computing*, 2004.

[27] Victor Cohen and Axel Parmentier. Two generalizations of Markov blankets. *arXiv:1903.03538*, 2019.

[28] A.J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand. *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Springer, 2006.

[29] Michele Conforti, Gerard Cornuejols, and Giacomo Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[30] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393 – 405, 1990.

[31] Alexandra Coppe, Matthew J. Pais, Raphael T. Haftka, and Nam H. Kim. Using a simple crack growth model in predicting remaining useful life. *Journal of Aircraft*, 49:1965–1973, 2012.

[32] Cassio P. de Campos and Fabio Gagliardi Cozman. Inference in credal networks through integer programming. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 145–154, 2007.

[33] Cassio P. de Campos and Qiang Ji. Strategy selection in influence diagrams using imprecise probabilities. *arXiv:1206.3246*, 2012.

[34] Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51:850–865, 2003.

[35] Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Morgan & Claypool Publishers, 2013.

[36] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58:595–612, 2010.

[37] Brian T. Denton. *Optimization of Sequential Decision Making for Chronic Diseases: From Data to Decisions*, chapter 13, pages 316–348. 2018.

[38] F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10:98–108, 1963.

[39] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63:68–77, 2019.

[40] Yann Dujardin, Tom Dietterich, and Iadine Chades. α-min: A compact approximate solver for finite-horizon POMDPs. In *International Joint Conferences on Artificial Intelligence*, pages 2582–2588, 2015.

[41] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59:295–320, 2017.

[42] James E. Eckles. Optimum maintenance with incomplete information. *Operations Research*, 16:1058–1067, 1968.

[43] Maxim Egorov, Zachary N. Sunberg, Edward Balaban, Tim A. Wheeler, Jayesh K. Gupta, and Mykel J. Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research*, 18:1–5, 2017.

[44] Hugh Ellis, Mingxiang Jiang, and Ross B. Corotis. Inspection, maintenance, and repair with partial observability. *Journal of Infrastructure Systems*, 1:92–99, 1995.

[45] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Mathematical Programming*, 171:115–166, 2018.

[46] Daniela P. De Farias and Benjamin V. Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51:2003, 2001.

[47] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Exploration Newsletter*, 15:1–10, 2014.

[48] Arthur M. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2, 1974.

[49] J. C. Gittins and D. M. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66:561–565, 1979.

[50] John C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley, 1989.

[51] Medjaher K. Gouriveau, R. and N. Zerhouni. *PHM and Predictive Maintenance*, chapter 1, pages 1–13. 2016.

[52] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019.

[53] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[54] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13, 2000.

[55] Jeffrey Thomas Hawkins. *A Lagrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. PhD thesis, Massachusetts Institute of Technology, 2003.

[56] Ronald A. Howard and James E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume II, pages 721–762. Strategic Decisions Group, 1984.

[57] Ronald A Howard and James E Matheson. Influence diagrams. *Decision Analysis*, 2:127–143, 2005.

[58] Barry Hurley, Barry O'Sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, and Simon De Givry. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21:413–434, 2016.

[59] Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Proceedings of the Seventh International Conference on Neural Information Processing Systems*, page 345–352, 1994.

[60] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20:1483 – 1510, 2006.

[61] Kamran Javed, Rafael Gouriveau, Ryad Zemouri, and Noureddine Zerhouni. Improving data-driven prognostics by assessing predictability of features. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, pages 555–560, 2011.

[62] Frank Jensen, Finn V Jensen, and Søren L Dittmer. From influence diagrams to junction trees. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 367–373, 1994.

[63] Marine Jouin, Rafael Gouriveau, Daniel Hissel, Marie-Cécile Péra, and Noureddine Zerhouni. Prognostics and Health Management of PEMFC - state of the art and remaining challenges. *International Journal of Hydrogen Energy*, 38:15307–15317, 2013.

[64] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[65] Arthur B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5: 558–562, 1962.

[66] Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166:165–193, 2005.

[67] Kesav Kaza, Rahul Meshram, Varun Mehta, and Shabbir N. Merchant. Sequential decision making with limited observation capability: Application to wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 5:237–251, 2019.

[68] Arindam Khaled, Eric A. Hansen, and Changhe Yuan. Solving limited-memory influence diagrams using branch-and-bound search. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2013.

[69] Jong Woo Kim, Go Bong Choi, and Jong Min Lee. A POMDP framework for integrated scheduling of infrastructure maintenance and inspection. *Computers & Chemical Engineering*, 112:239–252, 2018.

[70] Michael Jong Kim and Viliam Makis. Optimal control of a partially observable failing system with costly multivariate observations. *Stochastic Models*, 28:584–608, 2012.

[71] Michael Jong Kim and Viliam Makis. Joint optimization of sampling and control of partially observable failing systems. *Operations Research*, 61:777–790, 2013.

[72] Michael Jong Kim, Rui Jiang, Viliam Makis, and Chi-Guhn Lee. Optimal Bayesian fault prediction scheme for a partially observable system subject to random failure. *European Journal of Operational Research*, 214:331 – 339, 2011.

[73] Michael Jong Kim, Viliam Makis, and Rui Jiang. Parameter estimation for partially observable systems subject to random failure. *Applied Stochastic Models in Business and Industry*, 29:279–294, 2013.

[74] Anton J. Kleywegt, Vijay S. Nori, and Martin W. P. Savelsbergh. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36:94–118, 2002.

[75] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[76] Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and economic behavior*, 45:181–221, 2003.

[77] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. PAC reinforcement learning with rich observations. In *Proceedings of the Twenty-Ninth Advances in Neural Information Processing Systems*, 2016.

[78] Vikram Krishnamurthy. *Partially observed Markov decision processes: From filtering to controlled sensing*. Cambridge University Press, 2016.

[79] Vikram Krishnamurthy and Bo Wahlberg. Partially observed Markov decision process multiarmed bandits—structural results. *Mathematics of Operations Research*, 34:287–302, 2009.

[80] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of the Fourth Conference on Robotics: Science and Systems*, 2008.

[81] Steffen L Lauritzen and Dennis Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[82] Thanh Trung Le, Florent Chatelain, and Christophe Bérenguer. Hidden Markov Models for diagnostics and prognostics of systems under multiple deterioration modes. In *Proceedings of the Twenty-fourth Conference on European Safety and Reliability*, 2014.

[83] Junkyu Lee, Alexander T. Ihler, and Rina Dechter. Join graph decomposition bounds for influence diagrams. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 1053–1062, 2018.

[84] Yanjie Li, Baoqun Yin, and Hongsheng Xi. Finding optimal memoryless policies of POMDPs under the expected average reward criterion. *European Journal of Operational Research*, 211:556 – 567, 2011.

[85] Leo Liberti. Introduction to global optimization. Technical report, 2008.

[86] Michael L. Littman. The witness algorithm: Solving partially observable Markov decision processes. Technical report, 1994.

[87] Michael L. Littman. Memoryless policies: Theoretical limitations and practical results. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3: From Animals to Animats 3*, pages 238–245, 1994.

[88] Qiang Liu. *Reasoning and Decisions in Probabilistic Graphical Models–A Unified Framework*. University of California, Irvine, 2014.

[89] Qiang Liu and Alexander Ihler. Belief propagation for structured decision making. In *Uncertainty in Artificial Intelligence*, pages 523–532, 2012.

[90] Viliam Makis. Multivariate Bayesian control chart. *Operations Research*, 56:487–496, 2008.

[91] Viliam Makis and Xiamei Jiang. Optimal replacement under partial observations. *Mathematics of Operations Research*, 28:382–394, 2003.

[92] Mohammad Marufuzzaman, Ridvan Gedik, and Mohammad S. Roni. A benders based rolling horizon algorithm for a dynamic facility location problem. *Computers & Industrial Engineering*, 98:462 – 469, 2016.

[93] Jirí Matouek and Bernd Gärtner. *Understanding and Using Linear Programming*. Springer, 2006.

[94] Denis D. Maua. Equivalences between maximum a posteriori inference in Bayesian networks and maximum expected utility computation in influence diagrams. *International Journal of Approximate Reasoning*, 68:211–229, 2016.

[95] Denis D. Mauá and Cassio P. Campos. Solving decision problems with limited information. In *Proceedings of the Twenty-fifth Conference on Advances in Neural Information Processing Systems*, pages 603–611, 2011.

[96] Denis D. Mauá and Fabio Gagliardi Cozman. Fast local search methods for solving limited memory influence diagrams. *International Journal of Approximate Reasoning*, 68:230–245, 2016.

[97] Denis D. Mauá, Cassio P. de Campos, and Marco Zaffalon. The complexity of approximately solving influence diagrams. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 604–613, 2012.

[98] Denis D. Mauá, Cassio P. de Campos, and Marco Zaffalon. Solving limited memory influence diagrams. *Journal of Artificial Intelligence Research*, 44:97–140, 2012.

[99] Denis D. Mauá, Cassio P. De Campos, and Marco Zaffalon. On the complexity of solving polytree-shaped limited memory influence diagrams with binary variables. *Artificial Intelligence*, 205:30–38, 2013.

[100] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i – convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[101] Varun Mehta, Rahul Meshram, Kesav Kaza, and Shabbir N. Merchant. Multi-armed bandits with constrained arms and hidden states. 2017.

[102] Varun Mehta, Rahul Meshram, Kesav Kaza, and Shabbir N. Merchant. Sequential decision making with limited observation capability: Application to wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 5:237–251, 2019.

[103] Adam J. Mersereau. Information-sensitive replenishment when inventory records are inaccurate. *Production and Operations Management*, 22:792–810, 2013.

[104] Rahul Meshram, D. Manjunath, and Aditya Gopalan. On the whittle index for restless multiarmed hidden Markov bandits. *IEEE Transactions on Automatic Control*, 63:3046–3053, 2018.

[105] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, page 165–172, 1998.

[106] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, page 417–426, 1999.

[107] George E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28:1–16, 1982.

[108] Guido Montufar, Keyan Ghazi-Zahedi, and Nihat Ay. Geometry and determinism of optimal stationary control in partially observable Markov decision processes. *arXiv:1503.07206*, 2015.

[109] Eija Myötyri, Urho Pulkkinen, and Kaisa Simola. Application of stochastic filtering for lifetime prediction. *Reliability Engineering & System Safety*, 91:200–208, 2006. Selected Papers Presented at QUALITA 2003.

[110] Dennis Nilsson and Michael Höhle. Computing bounds on expected utilities for optimal policies based on limited information. Technical Report 94, Danish Informatics Network in the Agriculture Sciences, 2001.

[111] José Niño-Mora. Dynamic allocation indices for restless projects and queueing admission control: a polyhedral approach. *Mathematical Programming*, 93:361–413, 2002.

[112] Katerina P. Papadaki and Warren B. Powell. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Research Logistics*, 50:742–769, 2003.

[113] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12:441–450, 1987.

[114] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24:293–305, 1999.

[115] Paul Paris and Fazil Erdogan. A Critical Analysis of Crack Propagation Laws. *Journal of Basic Engineering*, 85:528–533, 1963.

[116] Mahshid Salemi Parizi and Archis Ghate. Weakly coupled Markov decision processes with imperfect information. In *Proceedings of the Winter Simulation Conference*, pages 3609–3602, 2019.

[117] Axel Parmentier, Victor Cohen, Vincent Leclère, Guillaume Obozinski, and Joseph Salmon. Integer programming on the junction tree polytope for influence diagrams. *INFORMS Journal on Optimization*, 2:209–228, 2020.

[118] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[119] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

[120] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1025–1030, 2003.

[121] Kim Leng Poh and Eric Horvitz. A graph-theoretic analysis of information value. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, pages 427–435, 1996.

[122] Warren B. Powell. *Clearing the Jungle of Stochastic Optimization*, chapter 4, pages 109–137.

[123] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2nd edition, 2011.

[124] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[125] Lawrence R. Rabiner and Biing H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 1986.

[126] Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv:1908.05659*, 2019.

[127] Jørgen Glomvik Rakke, Magnus Stålhane, Christian Rørholt Moe, Marielle Christiansen, Henrik Andersson, Kjetil Fagerholt, and Inge Norstad. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C: Emerging Technologies*, 19:896 – 911, 2011.

[128] Alistair R.Clark. Rolling horizon heuristics for production planning and set-up scheduling with backlogs and error-prone demand forecasts. *Production Planning & Control*, 16: 81–97, 2005.

[129] Neil Robertson and Paul D. Seymour. Graph minors. V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92 – 114, 1986.

[130] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[131] Natarajan Sakthivel, V. Sugumaran, and S. Babudevasenapati. Vibration based fault diagnosis of monoblock centrifugal pump using decision tree. *Expert Systems with Applications*, 37:4040–4049, 2010.

[132] A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher. Metrics for evaluating performance of prognostic techniques. In *2008 International Conference on Prognostics and Health Management*, pages 1–17, 2008.

[133] Petra Scheffler. A linear algorithm for the pathwidth of trees. In *Topics in combinatorics and graph theory*, pages 613–620. 1990.

[134] W. G. Schneeweiss. Fault-tree analysis using a binary decision tree. *IEEE Transactions on Reliability*, 34:453–457, 1985.

[135] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

[136] Suresh Sethi and Gerhard Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29:387–416, 1991.

[137] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[138] Ross D. Shachter. Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 480–487, 1998.

[139] Glenn Shafer and Prakash P. Shenoy. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–351, 1990.

[140] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27:1–51, 2013.

[141] Prakash P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations research*, 40:463–484, 1992.

[142] Xiao-Sheng Si, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. Remaining useful life estimation – a review on the statistical data driven approaches. *European Journal of Operational Research*, 213:1 – 14, 2011.

[143] Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, pages 284–292, 1994.

[144] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[145] Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 520–527, 2004.

[146] Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 542–549, 2005.

[147] Edward J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304, 1978.

[148] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1, 2011.

[149] David Sontag, Do Kook Choe, and Yitao Li. Efficiently searching for frustrated cycles in MAP inference. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 795–804, 2012.

[150] Weixiang Sun, Jin Chen, and Jiaqing Li. Decision tree and PCA-based fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 21:1300 – 1317, 2007.

[151] Diego Alejandro Tobon-Mejia, Kamal Medjaher, Noureddine Zerhouni, and Gerard Tripot. A data-driven failure prognostics method based on mixture of gaussians hidden Markov models. *IEEE Transactions on Reliability*, 61:491–503, 2012.

[152] Huseyin Topaloglu. Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research*, 57:637–649, 2009.

[153] Kwok-Leung Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, and Wenbin Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015:1–17, 2015.

[154] Ciriaco Valdez-Flores and Richard M. Feldman. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics*, 36: 419–446, 1989.

[155] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[156] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[157] Erwin Walraven and Matthijs Spaan. Column generation algorithms for constrained POMDPs. *The Journal of Artificial Intelligence Research*, 62:489–533, 2018.

[158] Erwin Walraven and Matthijs T. J. Spaan. Point-based value iteration for finite-horizon POMDPs. *Journal of Artificial Intelligence Research*, 65:307–341, 2019.

[159] Hongzhou Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139:469 – 489, 2002.

[160] Mei Wang and Jie Wang. Chmm for tool condition monitoring and remaining useful life prediction. *The International Journal of Advanced Manufacturing Technology*, 59:463–471, 2012.

[161] Yiwei Wang, Christian Gogu, Nicolas Binaud, Christian Bes, and Jian Fu. A model-based prognostics method for fatigue crack growth in fuselage panels. *Chinese Journal of Aeronautics*, 32:396 – 408, 2019.

[162] P. Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25:287–298, 1988.

[163] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38:153–183, 2013.

[164] John K. Williams and Satinder P. Singh. Experimental results on learning stochastic memoryless policies for partially observable Markov decision processes. In *Proceedings of the Eleventh Conference on Advances in Neural Information Processing Systems*, pages 1073–1080. 1999.

[165] Fan Ye, Helin Zhu, and Enlu Zhou. Weakly coupled dynamic program: Information and Lagrangian relaxations. *IEEE Transactions on Automatic Control*, 63:698–713, 2018.

[166] Changhe Yuan, Xiaojian Wu, and Eric Hansen. Solving multistage influence diagrams using branch-and-bound search. In *Proceedings of the Twenty-sixth Conference on Uncertainty in Artificial Intelligence*, pages 691–700, 2010.

[167] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, and et al. Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59: 56–65, 2016.

[168] Nevin L. Zhang and Wenju Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical report, The Hong Kong University of Science and Technology, 1996.