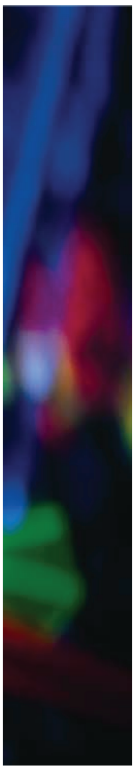
modèles et méthodes pour le génération de processus de fabrication reconfigurables

Qing Xia

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech



2017-ENAM-0004

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité "Génie Industriel "

présentée et soutenue publiquement par

Qing XIA

le 3 Mars 2017

Modèles et méthodes pour la génération de processus de fabrication reconfigurables

Directeur de thèse : **Jean-Yves DANTAN**
Co-encadrement de la thèse : **Alain ETIENNE, Ali SIADAT**

Jury

M. Michel ALDANONDO, Professeur, Mines Albi - IMT, Université de Toulouse	Président / Rapporteur
M. François VILLENEUVE, Professeur, G-SCOP, Université Grenoble Alpes	Rapporteur
Mme. Lihong QIAO, Professeur, Intel. Manufg. Tech. & Sys. Lab., Beihang University	Rapporteur
M. Nabil ANWER, Maître de conférence, IUT de Saint Denis, Université Paris XIII	Examinateur
M. Jean-Yves DANTAN, Professeur, LCFC, Arts et Métiers ParisTech, Metz	Examinateur
M. Alain ETIENNE, Maître de conférence, LCFC, Arts et Métiers ParisTech, Metz	Examinateur
M. Ali SIADAT, Professeur, LCFC, Arts et Métiers ParisTech, Metz	Examinateur
M. Aamer BAQAI, Associate Professor, National University of Sciences & Technology	Invité

T H È S E

Arts et Métiers ParisTech - Campus de Metz
Laboratoire de Conception Fabrication Commande

Avant Propos

Ce mémoire de thèse représente une synthèse de mes activités de recherche menées depuis octobre, 2013.

Ces activités ont été effectuées sous un statut :

- d'allocataire d'une bourse (China Scholarship Council (CSC), China) de septembre, 2013 à août, 2016. Cette bourse a été obtenue suite à un concours nationl, pour le développement des ressources humaines.

Ces travaux s'inscrivent dans la problématique de la génération de processus de production reconfigurables, ils visent particulièrement à répondre aux questions :

- Quelle est la définition de processus de fabrication reconfigurables?
- Comment représenter les informations hétérogènes et les connaissances nécessaires pour générer des processus de fabrication reconfigurables?
- Comment générer des processus reconfigurables pour une famille de produits/pièces?
- Comment appliquer les processus de fabrication reconfigurables pour gérer la complexité de fabrication induite par la variété de produits?

Etant donnée l'aspect international de cette thèse et suite à la décision DG2009-46 du 1er Octobre 2009, le Directeur Général d'Arts et Métiers ParisTech a autorisé la rédaction de ce mémoire en deux langues : anglaise et française. De ce fait, ce document se divise en deux parties : un résumé étendu en français sans figure et le descriptif des travaux en anglais avec les figures.

Foreword

This thesis represents a summary of my research activities preformed since October, 2013.

These activities were performed under a status:

- Recipient of a scholarship (China Scholarship Council (CSC), China) from September, 2013 to August, 2016. This scholarship was obtained after a national level selection process for the development of high level universities in China.

This work is done in the domain of manufacturing process planning; it attempts in particular to respond to the question: "What are the concepts, representation models and generation methods to support reconfigurable process planning for product/part variety".

While taking into account the international aspect of this work, the "Directeur général d'Arts et Métiers ParisTech" authorised the writing of the thesis report in two languages: English and French (Decision DG2009-46 October, 2009). Hence, this document is divided into two parts: an extended summary in French (without figures) and the detailed description of the research in English (with illustrations).

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my thesis director, Professor Jean-Yves DANTAN. Without his guidance, encouragement and inspiration, this work would not have been possible. His creative insights and systematic thoughts help me focus on the most valuable issues in the subject. I also appreciate the free academic ambience under his supervision.

I am also deeply grateful to my co-supervisor, Doctor Alain ETIENNE, who provided me strong support and guidance throughout the progress of this research work. I have learnt a lot from his responsible attitude towards teaching and research. I am also indebted to my co-supervisor, Professor Ali SIADAT, for his continuous support and constructive suggestions as well as helping me deal with a lot of administrative stuff. They work perfectly as a team with their students.

Many thanks to my committee members, Professor Michel ALDANONDO, Professor François VILLENEUVE, Professor Lihong QIAO and Doctor Nabil ANWER, for their time and invaluable comments. Special thank to Doctor Aamer BAQAI for being an invited jury member during my defence.

I would like to acknowledge all of my colleagues in LCFC for their support and encouragement. Special thanks to Antoine, Shirin, Etienne, Amir, Roozbeh, Lazhar, Mehrdad, Leyla, Ismail, Edo, Mohhamad, Uzair and Jelena, for bringing me the friendly and vibrant working atmosphere at B148. Meanwhile, my gratefulness also goes to my Chinese friends, Fan, Peng, Da, Ke, Zhongkai, Jinna, Jianjie, Zhicheng, Haitao, XianQiong, Yanfeng, Bin and Jianchang, for their support and company.

I would like to thank my beloved parents. This work would never be possible without their care, love, understanding and endless support.

At last, I truly appreciate China Scholarship Council, who provided the financial support for this work.

Contents

List of Figures

List of Tables

Acronym

AI	Artificial Intelligent	32
APP	Assembly Process Planning	19
BOM	Bill-of-Materials	22
BOMO	Bill-of-Materials-and-Operations	24
CAAP	Computer-Aided Assembly Process Planning	33
CAPP	Computer-Aided Process Planning	33
CNF	Conjunctive Normal Form	127
CP	Common Part	73
CPF	Common Part Family	73
CSP	Constraint Satisfaction Problem	144
DCSP	Dynamic Constraint Satisfaction Problem	144
DFC	Datum Flow Chain	70
DOF	Degree Of Freedom	61
FC	Feature Cluster	51
FM	Function Module	51
FOL	First Order Logic	122
FV	Feature Variant	51
GBOM	Generic Bill-of-Materials	23
GD&T	Geometric Dimension and Tolerance	30
MC	Mass Customization	5
MPP	Machining Process Planning	19
OPF	Optional Part Family	73
P3S	Product,Process and Production System	5

PP	Personalized Part	73
PVDA	Product Variety Decomposition Architecture	43
R-BF Algorithm	Resolution-based Breadth-First Algorithm	121
RAPP	Reconfigurable Assembly Process Plan	81
RMOP	Reconfigurable Machining Operation Plan	81
RMPP	Reconfigurable Machining Process Plan	81
RPP	Reconfigurable Process Plan	6
UML	Unified Modelling Language	21

Part I

Résume Étendu En Français

Introduction

Motivation

Aujourd'hui, les entreprises sont confrontées à un dilemme : D'une part, le nombre croissant de variantes de produit nécessite des processus de fabrication diversifiés pour gagner de nouveaux marchés. D'autre part le système de fabrication requiert des processus stables pour réaliser des économies d'échelle. Deux paradigmes de fabrication ont été proposés par la littérature pour faire face à ces deux principaux défis. L'un deux est la personnalisation de masse qui vise à offrir une variété de produits abordables avec une efficacité de production permettant de contrôler : le coût, la qualité et le délai de fabrication et de livraison ; un autre est la coévolution : du produit, de son processus de fabrication et de son système de production. Dans ce paradigme, les «changements» de fabrication sont traités par la configuration répétitf du produit, du processus et du système de production dans le temps.

La conception de familles de produits est une stratégie efficace pour ces deux paradigmes. Elle permet aux variantes de produits de partager un certain nombre de composants et de fonctions, tandis que chaque variante de produit peut avoir des spécifications uniques pour répondre aux exigences spécifiques du client. L'organisation de variantes de produit dans une famille peut non seulement diminuer la complexité de la conception pour la variété de produits données, mais également en faciliter la production. Toutefois, les avantages de la conception orientée famille de produits ne peuvent pas être garantis s'il n'existe pas de méthode appropriée pour générer les processus de fabrication des variantes dans la famille de produits.

Les méthodes classiques de génération de processus de fabrication se concentrent principalement sur la génération de processus pour un produit ou une pièce donné. Ces méthodes présentent les limites suivantes dans le cas de familles de produits :

- Elles considèrent rarement les changements futurs du modèle de famille de produits et des ressources disponibles ;
- Les processus générés sont susceptibles de devenir rapidement obsolètes pour les nouvelles variantes de produits ou de pièces;
- Elles nécessitent de lourds calculs pour regénérer les processus pour les nouvelles variantes de produits/pièces.

ElMaraghy (2007) propose une nouvelle méthode de génération des processus de fabrication, les « Processus de fabrication Reconfigurables (RPP) », afin de remédier aux limites des méthodes traditionnelles pour des variantes de produits. L'approche RPP adopte le concept de configuration dans l'activité de la génération des processus pour une famille de produits. Les variantes de processus pour les nouvelles variantes de produits sont générées en tenant compte des ressources de fabrication disponibles. L'approche RPP se montre plus intéressante que les méthodes traditionnelles dans les aspects de l'efficacité et de la variabilité. Il a été identifié comme l'un des principaux facteurs permettant la réalisation de la personnalisation de masse et de la coévolution du triptyque Produit/Processus/Production. Solution émergente, RPP est pourtant encore loin d'une utilisation industrielle concrète. Avant d'être appliquée dans l'industrie, cette approche doit montrer ses capacités à faire face aux scénarios suivants :

- Scénario I - Gestion de la variation de processus à différents niveaux de granularité de la conception du produit : niveaux de produit, niveaux de pièce et niveaux de ses composants géométriques et fonctionnels;
- Scénario II - Gestion de la variation de processus dans un système de fabrication agile.

Objectifs des travaux de recherche

Pour réaliser les deux scénarios cités précédemment, ces travaux de recherche visent à développer une méthodologie, une architecture, des modèles et des algorithmes appropriés, au moyen desquels les processus de fabrication pour toutes variantes de produit/pièce d'une famille peuvent être générés facilement et automatiquement.

Ses travaux cherchent à répondre aux questions suivantes :

- Quelle est la définition de RPP?
- Comment représenter les informations hétérogènes et les connaissances nécessaires pour générer des processus de fabrication reconfigurables?
- Comment générer des processus reconfigurables pour une famille de produits/pièces?
- Comment appliquer l'approche RPP pour gérer la complexité de fabrication induite par la variété de produits?

Organisation de la thèse

En se concentrant sur les objectifs de recherche définis précédemment, cette thèse est organisée en cinq chapitres. Après l'introduction, le premier chapitre se focalise sur la définition des concepts nécessaires au développement du sujet proposé, afin de mieux définir la portée de la thèse. Puis, il examine l'état de l'art des thématiques de recherche connexes, y compris les modèles produit orienté fabrication, la modélisation des configurations pour la variété des produits, et des configurations pour la variété des gammes de fabrication et leur génération.

Le chapitre 2 propose des méthodes de modélisation structurée pour représenter les informations de conception d'une famille de produits.

Le chapitre 3 propose des modèles de représentation des processus de fabrication reconfigurables. Dans ce chapitre, le concept de processus de fabrication reconfigurable est développé en deux niveaux de granularité : processus d'assemblage reconfigurables et processus d'usinage reconfigurables.

Le chapitre 4 traite de la modélisation des connaissances et des méthodes pour la génération des processus reconfigurables. Il répond à la question : « comment traiter les informations de conception représentées par les modèles proposés au chapitre 2 afin de générer les processus reconfigurables représentées par les modèles proposés au chapitre 3 ». La logique du premier ordre est utilisée dans ce chapitre pour modéliser les connaissances impliquées dans la gestion des processus reconfigurables. Les algorithmes de génération de processus reconfigurables, qu'ils soient d'assemblage, ou d'usinage, sont respectivement développés dans ce chapitre.

Le chapitre 5 est consacré aux applications de génération de processus de fabrication reconfigurables dans le cadre des paradigmes de la personnalisation de masse et de la

coévolution du triptyque Produit/Processus/Production. Une méthode intégrée de configuration de produit et de configuration du processus de fabrication reconfigurable est proposée pour répondre aux contraintes dynamiques des systèmes de production. Des objectifs d'optimisation et des méthodes d'évaluation pour générer les processus optimaux pour une variante de produit/pièce spécifique dans un scénario de fabrication spécifique sont étudiés. Des exemples illustratifs sont présentés dans ce chapitre pour montrer l'efficacité et la faisabilité des méthodes proposées.

Dans la conclusion, les contributions et les limites des méthodes proposées dans cette thèse sont soulignées permettant d'ouvrir aux perspectives et travaux futurs.

Chapitre 1

Définitions

Ce chapitre commence par les principaux définitions des concepts de l'ingénierie de fabrication afin de mieux définir la portée de ces travaux. Ensuite, une revue de la littérature sur la modélisation des produits pour la fabrication, la technologie de configuration et la génération des processus de fabrication est proposée. Le chapitre se conclue par une évaluation de la littérature en insistant sur la pertinence de ce travaux de recherche.

1.1 Définitions des concepts principaux

Puisque l'objectif de cette recherche est de développer des modèles de représentation et des méthodes pour la génération de processus reconfigurables pour une famille de produits, les concepts de produits, de pièces, de familles de produits, de familles de pièces, sont clairement définis afin de montrer la portée de cette recherche.

Définition 1(*Produit*). Un produit est un assemblage mécanique fabriqué par une séquence d'opération d'assemblage qui fixent progressivement tous les composants de ce produit ensemble. Un produit fonctionne individuellement pour répondre aux exigences de certains clients.

Définition 2(*Pièce*). Une pièce est un composant inséparable lors de l'assemblage d'un produit. Une pièce est le plus petit bloc de construction d'un produit et chaque composante d'un produit est une pièce ou une composition d'un ensemble de pièces.

Définition 3(*Variété des produits*). La variété des produits décrit un domaine de conception pour des produits similaires. Ces produits similaires sont définis par un certain nombre de variables de conception communes et de variables de conception personnalisées. Les

variables de conception communes décrivent les caractéristiques communes de ces produits similaires tandis que les variables de conception personnalisées définissent leurs caractéristiques individualisées. La valeur des variables de conception peuvent être un ensemble fini ou un infini.

Définition 4(*Famille de produits*). Une famille de produits représente un domaine technique pour des assemblages similaires, qui est ensuite décomposé en ensembles d'architecture et d'attributs. L'architecture limite l'éventail des composantes variétales utilisées pour construire la structure fonctionnelle et physique d'un assemblage, tandis que les ensembles d'attributs limitent les valeurs d'attributs des composantes. Une variante d'assemblage est dérivée en choisissant un ensemble de composants dans des ensembles d'architecture et les valeurs pour les attributs de ces composants dans des ensembles d'attributs correspondants.

Définition 5(*Variété des pièces*). La variété de pièce décrit un domaine de conception pour des pièces similaires qui sont définies par un certain nombre de variables de conception communes et de variables de conception personnalisées. Les variables de conception communes décrivent les caractéristiques communes de ces pièces similaires tandis que les variables de conception personnalisées définissent leurs caractéristiques individualisées. La valeur des variables de conception peuvent être un ensemble fini ou non.

Définition 6(*Famille de pièces*). Une famille de pièces représente un domaine de pièces qui est ensuite décomposé en une architecture et un ensemble d'attributs. Une variante de pièce peut être définie par choisir des composants à partir de l'architecture et les valeurs pour les attributs de ces composants dans des ensembles d'attributs correspondants.

Définition 7(*Génération des processus de fabrication*). La génération des processus de fabrication consiste en une séquence d'activités qui définissent en détail le processus qui transforme la matière première en la forme désirée. Les activités impliquées comprennent: l'interprétation des spécifications de conception, la sélection des procédés de fabrication, la sélection des outils de fabrication, la détermination des paramètres des processus de fabrication, la détermination des séquences d'opération et le calcul des coûts.

Différents procédés de fabrication demandent des approches différentes pour la génération des processus de fabrication. Cette recherche se limite à deux procédés de fabrication: assemblage et usinage. Donc, elle ne se consacre qu'aux questions liées à la génération des processus d'assemblage et génération des processus d'usinage reconfigurables.

1.2 Bibliographie

1.2.1 Modélisation de produit orienté fabrication

La modélisation des produits orientée les questions relatives aux informations sur les produits qui sont nécessaires pour leur fabrication et à la façon dont elles peuvent être représentées de manière informatisée. L'importance de la modélisation des produits a été largement reconnue par les communautés de recherche des dernières décennies. De nombreux modèles produits ont été développés pour supporter l'assemblage intégré par ordinateur et l'usinage intégré par ordinateur. Ces modèles incluent :

- Modèles basés sur la nomenclature (Garwood, 1988; Hegge et Wortmann, 1991; Jiao et al., 2000)

- Modèles basés sur les graphes des liaison (Henrioud and Bourjault, 1992 ; de Mello et Sanderson, 1991a ; De Lit et Delchambre, 2003)

- Modèles basés sur des fonctions (Shah, 1991a,b; Allada and Anand, 1995 ; Case and Wan Harun, 2000; Shah and Rogers, 1993; Mascle, 2002; Li et al., 2010; Zhang et al., 2015 ; Catania, 1991; Gu, 1994; Gonzalez and Rosado, 2004; Amaitik and Kilic, 2005; Li et al. , 2006b)

Les données produit représentées dans les modèles basés sur la nomenclature ne conviennent que pour la planification matérielle, mais les modèles basés sur des fonctions est une méthode efficace pour représenter les informations nécessaires d'une pièce pour la génération de son processus d'assemblage et de son processus d'usinage. Cependant, une difficulté pour l'application des approches existantes reste qu'il n'existe toujours pas de modèle universel pouvant être utilisé à la fois pour la génération de processus d'assemblage et d'usinage. Les chercheurs doivent donc developer leurs propres modèles en fonction de leurs scénarii d'application. En outre, la plupart des modèles fonctionnels existants sont conçus pour représenter les spécifications de conception d'une seule pièce; peu d'entre eux considèrent le concept de famille de pièces.

1.2.2 Technique de configuration pour la variété de produit et la variété de processus

La configuration est considérée comme une technologie efficace pour réduire les coûts de développement et de maintenance pour la mise en œuvre de la personnalisation de masse (Felfernig et al., 2014). La configuration de produit est une application réussie des techniques de configuration. La configuration de produit permet la génération rapide de nouvelles variantes de produit à partir d'un ensemble de composants de produits prédéfinis au lieu de les créer à partir de zéro (Salvador and Forza, 2004a; Zhang, 2014). De nombreuses techniques d'intelligence artificielle ont été appliquées dans la configuration du produit pour automatiser cette activité complexe (Yang et al., 2008; Hong et al., 2008).

Un effet immédiat de la variété de produit est la croissance du nombre de variantes de processus. Certains chercheurs ont appliquer les technique de configuration sur la génération de variantes de processus (Schierholt, 2001; Aldanondo and Vareilles, 2008; Jiao et al., 2004; Zheng et al., 2008; Pitiot et al., 2014; Wang et al., 2015).

Plusieurs travaux de recherches ont été consacrés à la technologie de configuration pour la configuration du produit, tandis que la technologie de configuration pour la génération de variantes de processus est moins utilisé. De plus, suite à l'étude des references il semble que la plupart des recherches existantes sur la configuration de processus choisissent une famille de produits comme leurs axes et aucune référence ne peut être trouvée sur la configuration de processus d'usinage pour une famille de pièces.

1.2.3 Génération de processus de fabrication

La génération de processus de fabrication définit toutes les étapes nécessaires pour guider le système de fabrication afin de générer la géométrie specifiée, les propriétés, la qualité d'une pièce ou d'un produit, dans les contraintes données tout en optimisant certains critères. Avec la technologie de l'information, l'approche manuelle pour la génération des processus d'assemblage et d'usinage est maintenant assistée par les systèmes d'informatiques (CAPP et CAAP).

De nombreuses technologies du domaine de l'informatique ont été utilisées pour automatiser les activités de génération des processus d'usinage, y compris : (1)Les technologies basées sur des fonctions (Sormaz and Khoshnevis, 2000;Wang et al., 2006; Givehchi and

Wang, 2015); (2)Technologies basées sur le connaissances (Liu and Wang, 2007; Denkena et al., 2007); (3) Technologies basées sur l'intelligence artificielle (Barrabes and Villeneuve, 1993; Qiao et al.,2000; Liu et al., 2013); (4) Technologies conformes à la norme (Xu et al., 2006; Chung and Suh, 2008); and (5)Technologies de l'Internet (Agrawal et al., 2009; Wang, 2013; Hu et al., 2008).

Certains chercheurs considèrent l'impact d'une variété de pièces dans leurs approches pour la génération des processus d'usinage. Mäntylä and Sohlenius (1993) ont développé un modèle de famille de pièces basé sur des fonctions qui sont utilisées pour contenir les spécifications du processus d'usinage.

Une ancienne approche pour générer un processus d'usinage pour une nouvelle variante de pièce est la réutilisation des processus existants, tels que l'approche basée sur la technologie de groupe (Burbidge, 1993) et l'approche basée sur le raisonnement à partir de cas. Cependant, ce type d'approches est le manque de flexibilité et d'adaptabilité en termes de nouveaux changements de conception sur des variantes partielles. Afin d'améliorer l'ancienne approche pour supporter la génération de processus pour une famille de produits, ElMaraghy (2006) a introduit une nouvelle approche: la génération de processus reconfigurable (RPP). Azab et al. (2007) a proposé une approche hybride pour configurer le processus existant afin de satisfaire les exigences d'usinage d'une nouvelle variante de pièces.

Sous l'aspect de la génération de processus d'assemblage assistée par ordinateur (CAAP), divers modèles de représentation de processus d'assemblage ont été présentés dans la littérature (Jiménez, 2013). En général, les modèles de processus d'assemblage peuvent être classés en deux catégories: modèles explicites et modèles implicites (de Mello et Sanderson, 1991b). Les représentations explicites expriment explicitement les tâches d'assemblage et la précédence entre celles-ci. Les modèles explicites comprennent : la liste d'ordre, l'arbre binaire (Wolter, 1991), le graphe orienté (de Mello et Sanderson, 1991b) et le graphe ET / OR (Thomas et al., 2003). Les modèles implicites sont basés sur les conditions d'établissement et sur les relations de précédence qui limitent implicitement les séquences d'assemblage réalisables.

De nombreux algorithmes ont été développés pour la génération de processus d'assemblage réalisables et optimale basées sur un modèle de séquence d'assemblage spécifique. Ces méthodes sont normalement établies sur un modèle de processus d'assemblage spécifique. Un moyen direct est d'énumérer tous les sous-ensembles possibles, puis de tester les possibilités

de toutes les séquences possibles entre ces sous-ensembles(de Mello and Sanderson, 1991a). Certains chercheurs utilisent des méthodes de désassemblage pour en déduire pars inversion des séquences d'assemblage inversement (Martinezet al., 2009; Li et al., 2014). En comparaison avec les méthodes directes, un avantage des méthodes de désassemblage est qu'elles ne seront jamais sans issue.

Afin d'améliorer l'efficacité de la résolution des problèmes, de nombreux chercheurs ont utilisé des algorithmes heuristiques pour générer des séquences d'assemblage telles que les algorithmes génétiques (Marian et al., 2006), les algorithmes d'essaims particulaires (Wang et Liu, 2010), les algorithmes de colonies de fourmis (Wang et al., 2005) et des réseaux neuronaux artificiels (Chen et al., 2008).

En comparaison avec la recherche sur la CAAP pour une seule assemblée, peu de travaux de recherche sont consacrés aux approches de la génération des processus d'assemblage pour une famille de produits. Une caractéristique commune de ces recherches existantes sur la CAAP pour une famille de produits est que les modèles génériques sont définis pour représenter l'universalité des membres de la famille de produits. Les processus d'assemblage pour une variante de produit particulière sont générés par l'analyse ou la modification de ces modèles génériques.

1.3 Conclusions

Dans ce chapitre, pour définir les bornes de cette thèse, on donne d'abord les définitions des concepts pertinents. Un concept de «domaine» est introduit pour étendre les définitions de famille de produits et de famille de pièces. Avec cette extension, une famille de produits/pièces peut avoir un nombre infini de variantes.

Parmi les modèles de produits existants orientés fabrication, le graphe des liaisons et les modèles basés sur les fonctions sont les modèles de représentation considérés plus efficaces que les modèles de représentation basés sur la nomenclature pour la génération des processus. Mais, la plupart des modèles existants se concentrent sur la représentation de l'information d'une seule pièce, un modèle de fonctions pour un ensemble de variantes de pièces similaires dans une famille de produits est rarement considéré dans la littérature.

En comparaison avec la configuration de produit, peu de recherches ont été effectuées sur la configuration du processus; seulques articles sur la configuration du processus se réfèrent

à la configuration des processus de fabrication.

Les recherches actuelles sur la génération des processus d'usinage et des processus d'assemblage sont effectuées séparément. Peu de recherches cherchent aux intégrer. En outre, bien que quelques pionniers essaient d'étudier la génération des processus de fabrication pour la variété de produits/parties, la plupart des méthodes existantes ne s'appliquent que pour un seul produit/pièce.

Un nouveau concept pour la génération des processus de fabrication: « La génération des processus reconfigurables » est proposée afin de gérer les processus pour l'évolution des familles de produits/pièces. Comme technologie émergente combinant à la fois la génération des processus de fabrication et la technologie de configuration, elle est identifiée comme un catalyseur clé pour le paradigme de la fabrication reconfigurable. Cependant, il n'existe aucune définition formelle, modèle de représentation ou cadre d'application pour la génération des processus reconfigurables, ce qui la rend encore loin de l'application industrielle. La nécessité d'une méthode de génération des processus axée sur la famille de produits/pièces et l'insuffisance des recherches favorisent la motivation de ces travaux de thèse.

Chapitre 2

Modèles de variétés de produits basé sur des entités

Ce chapitre propose des modèles de famille de produits et de pièces pour représenter des spécifications de conception nécessaires à la génération de processus de fabrication reconfigurable. Deux types d'informations sont modélisés dans ces modèles : Informations relatives à la configuration et informations relatives au processus. Une architecture de décomposition est proposée pour représenter l'architecture fonctionnelle et l'architecture physique au niveau du produit et du niveau de la pièce. Les contraintes de configuration entre des éléments reconfigurables dans l'architecture de décomposition sont exprimées en utilisant un schéma de représentation basé sur la logique propositionnelle. Les informations relatives aux processus, y compris les informations de jonction liées à l'assemblage d'une famille de produits et les spécifications de conception liées à l'usinage des familles de pièces à l'intérieur de la famille de produits, sont représentées par des modèles basé sur des entités.

2.1 Proposition d'un modèle produit

2.1.1 Architecture produit

L'architecture produit est une partie de l'architecture de décomposition d'une variété de produits proposée. La structure fonctionnelle définie dans le modèle produit constitue le premier niveau de l'structure de décomposition. Une structure fonctionnelle est une disposition hiérarchique d'entités fonctionnelles modulaires définies à partir des exigences fonc-

tionnelles. La structure fonctionnelle d'une variété de produits dans le modèle est une union des structures fonctionnelles de toutes les variantes des produits.

Une autre partie du modèle produit est la structure physique des variantes des produits. Comme la structure fonctionnelle, la structure physique est une union des structures physiques de toutes les variantes des produits. Donc, les variantes des produits pourraient partager les mêmes composants physiques, ils pourraient également avoir certains composants physiques personnalisés. En considérant toutes les situations possibles pour les composants physiques dans la structure physique, les composants physiques sont classés dans les quatre catégories suivantes:

- Pièces communes. Elles sont les pièces partagées par toutes les variantes des produits et leurs spécifications de conception restent les mêmes dans toutes les variantes des produits;

- Variantes des pièces dans une famille de pièces communes. Elles fournissent des fonctions communes partagées par toutes les variantes des produits, mais elles peuvent avoir des spécifications de conception différentes. La famille de pièces pour ces variantes des pièces est appelée une famille de pièces communes.

- Variantes des pièces dans une famille de pièces en option. Elles réalisent certaines fonctions facultatives partagées seulement par une partie de variantes des produits; la famille des pièces pour ces variantes des pièces est appelée une famille de pièces facultatives;

- Pièces personnalisées. Elles fournissent des fonctions exclusives pour une variante des produits spécifiques ou un ensemble de variantes des produits spécifiques et leurs spécifications de conception restent les mêmes dans la(les) variante(s) de produit(s) spécifique(s).

La structure physique d'une famille de produit dans l'architecture de décomposition représente l'arrangement les quatre types de composants physiques. Les variantes de parties sont représentées par leur famille de pièces.

2.1.2 Proposition d'opérateurs logiques pour représenter des relations de configuration

Trois types d'opérateurs logiques sont proposés pour représenter les relations de configuration entre les entités d'une variété dans différents niveaux de décomposition de l'architecture:

- Opérateur **AND**. Tous les composants physiques connectés à un opérateur **AND** doivent apparaître dans la même variante de produit;

- Opérateur **XOR**. Les composants physiques connectés à un opérateur **XOR** sont exclusifs l'un de l'autre, ce qui signifie qu'un seul d'entre eux peut apparaître dans une variante de produit.

- Opérateur **OPTION**. Le composant physique connecté à un opérateur OPTION est facultatif pour une variante de produit.

2.1.3 Proposition de représentation des familles de pièces

Dans le modèle produit, la structure physique des variantes des produits est représentée par l'organisation des pièces de ces variantes des produits en quatre catégories de pièces (pièce communes, famille de pièces communes, famille de pièces facultatives et pièces personnalisées), puis en les reliant via trois types d'opérateurs logiques en fonction de leurs relations de configuration dans les variantes des produits. Étant différentes avec des pièces communes et des pièces personnalisées dont les spécifications de conception restent les mêmes, les familles de pièces se composent de variantes des pièces similaires dont les spécifications de conception changent pour certaines instances. Par conséquent, le modèle produit doit être en mesure de représenter la variété et la configuration d'une famille de pièces.

Afin de profiter des points communs entre les variantes des pièces, au lieu de représenter chaque variante des pièce individuellement, nous proposons un modèle de variété des pièces. Dans ce modèle, la variété de pièces est divisée en trois niveaux de décomposition :

- Niveau des modules fonctionnels. Les fonctions de conception d'une pièce remplissent des fonctions techniques, ou des fonctions d'assemblage. Toutes les fonctions techniques des variantes des fonctions dans une famille de produits sont structurées dans ce niveau.

- Niveau des regroupements des entités. Un regroupement d'entité se rapporte à un ensemble de entités similaires appartenant au même type d'entité et réalisant les mêmes fonctions de conception sur les différentes variantes des pièces. Une entité dans un regroupement d'entité est appelées une instance d'entité. Toutes les instances d'entité héritent du même ensemble d'attributs, mais les valeurs de ces attributs peuvent être différentes.

- Niveau des variantes des entités . Le niveau des variantes des entités est le niveau bas du modèle de variété des pièces. Chaque instance d'entité est un conteneur d'information modulaire qui contient une partie des spécifications de conception d'une variante des pièces. Elle possède une structure orientée objet dans laquelle les attributions géométriques et techniques de chaque instance d'entité peuvent être instanciées à partir des entités génériques.

Dans un modèle de variété de pièces, les trois types d'opérateurs logiques sont également applicables pour décrire les relations de configuration parmi les composants de variété liées au même composante de niveau supérieur.

2.2 Proposition d'un schéma de représentation pour des relations de configuration

Dans le modèle produit, les composantes d'une famille de produits sont structurées en utilisant les trois types d'opérateurs logiques. Toutefois, les opérateurs logiques ne représentent que les relations de configuration entre les composants enfants du même composant parent à partir du niveau de décomposition supérieur. Donc, un schéma basé sur la logique propositionnelle pour donner une représentation universelle des contraintes de configuration entre les composants de variété dans le modèle produit est proposé dans cette section.

Le schéma proposé est utilisé pour représenter des contraintes de configuration dans les cas suivants :

- Des relations de configuration entre les composants enfants d'une même composante de niveau supérieur;

- Des relations de configuration entre les composants enfants des différents composants de niveau supérieur;

- Des contraintes de configuration sur les attributs des composants;
- Des relations de configuration entre des attributs et des composants.

2.3 Proposition d'un modèle produit

Le modèle produit doit également tenir compte des informations de conception de processus pour supporter la génération automatique de processus reconfigurables pour une famille de produits/pièces. Au niveau du produit, le modèle doit être capable de représenter les spécifications de conception pour la génération de processus d'assemblage; au niveau de la pièce, le modèle doit représenter les informations nécessaires à la génération de processus d'usinage. De plus, comme une famille de produits/pièces constituée d'un groupe de variantes, les points communs des informations liées aux processus parmi ces variantes doivent être traités efficacement dans le modèle afin d'obtenir une plus faible redondance des données.

2.3.1 Représentation des informations relatives aux processus pour une variété de pièces

Le modèle de représentation proposé pour une variété de pièces est établi via la technique de modélisation basée sur les entités. Dans la technique de modélisation, chaque entité est une entité modulaire instanciée à partir d'une entité générique. Une entité générique définit les spécifications géométriques d'une classe d'entité, y compris les éléments géométriques de l'entité, l'orientation ainsi que les formes, les relations topologiques et les attributs de ces éléments géométriques.

Afin de representer des spécifications des regroupements des entités qui sont les composants du modèle produit pour une famille de pièces, deux types de données sont adoptés pour exprimer les valeurs d'attribut d'un regroupement d'entité :

- Intervalle, pour les attributs qui prennent des valeurs continues;
- Ensemble, pour les attributs qui prennent des valeurs discrètes.

Un regroupement d'entité est une collection de variantes des entités similaires instanciées à partir de la même entité générique. La description d'un regroupement d'entité est une union de descriptions des variantes dans ce regroupement d'entité. Par conséquent, les

valeurs de même attribut des variantes sont rassemblées dans un intervalle ou un ensemble de façon à former les valeurs de l'attribut correspondant du regroupement d'entité.

2.3.2 Représentation des interactions entre les composantes de la variété

Interactions entre des entités

Une approche de représentation basée sur la connaissance est utilisée pour modéliser les interactions entre des entités d'une variante pièce. Dans cette approche, des prédicats représentent les types de base des objets dans le domaine des connaissances. Les prédicats N-ary expriment les relations entre les objets. Ces relations pourraient être les relations d'attributs, les dépendances de tolérance/référence et les interactions topologiques.

Lorsque l'on considère les interactions des entités au niveau de la famille de pièces, la situation devient compliquée. Une interaction des entités pourrait être la même entre tous les variantes des entités correspondantes. Par conséquent, afin de maximiser les avantages du concept de regroupement, les interactions des entités d'une famille de pièces sont représentées au niveau des regroupements d'entités, tandis que la diversité des interactions des entités est maintenue au niveau des variantes des entités.

Interactions entre des pièces

La représentation des relations de jonction entre toutes les pièces d'un produit est importante pour déterminer les processus d'assemblage pour un produit. Au niveau d'une famille de produit qui contient un ensemble de variantes, il pourrait y avoir les mêmes relations de jonction entre les pièces des différentes variantes du produit. Donc, afin d'éviter de représenter de manière répétée les informations de jonction commun parmi les pièce accouplées des variantes produit, un mécanisme de représentation basé sur les graphes des contacts est proposé. Le mécanisme de représentation intrése les informations de jonction entre les pièce accouplées à deux niveaux :

- Niveau de la famille de produit. Les informations de jonction représentées à ce niveau sont partagées par toutes les variantes des produits.
- Niveau de la variante du produit. Les informations de jonction représentées à ce niveau sont liées à des variantes de produit particulières.

Comme il est mentionné à la sous-section 2.1.1, dans le modèle produit, les composants physiques d'une famille de produits sont classées en quatre types : pièces communes, variantes dans une famille de pièces communes et variantes dans une famille de pièces en option et pièces personnalisées. Compte tenu des interactions entre les pièce de ces quatre types, 10 situations possibles ont été identifiées. Mais, ces situations de jonction ne peuvent pas être représentées toutes au niveau de la famille de produits. Donc, les règles visant à déterminer les situations de jonction privilégiées qui peuvent être représentées au niveau de la famille de produits sont proposées.

Avec les règles définies, le modèle Graphe des contacts est étendu pour tenir compte de l'information des interactions entre des pièces de toute la famille de produits. Les relations de jonction communes au niveau de la famille de produits sont représentées par les modèles au niveau de la famille de produits, tandis que les relations de jonction variables sont représentées par les modèles au niveau de la variante des produits.

2.4 Conclusions

Dans ce chapitre, des modèles de variétés de produits basés sur le concept d'entité sont proposés. Deux principes sont utilisés lors du développement de ces modèles de représentation: 1) décomposer de façon modulaire toute la variété de produits en modules configurables et utiliser les entités comme représentation atomiques; 2) représenter au maximum les similaritté entre toutes les variantes de la famille afin de réduire la redondance des données.

Deux aspects de l'information sur la variété de produits sont représentés dans les modèles proposés: information relative à la configuration et information relative au processus. Pour la représentation d'informations relatives à la configuration, on présente un modèle produit de variétés de produits, dans lequel la structure d'une famille de produits se décompose en différents niveaux de composantes variées: composants fonctionnels, familles de pièces, regroupements d'entites et variantes des entités. Les composants de variété sont liés en utilisant trois types d'opérateurs logiques. Un schéma de représentation basé sur la logique est défini pour décrire les contraintes de configuration entre les composants de variété.

Pour la représentation d'informations relatives au processus, une approche de représentation basée sur la connaissance est utilisée pour décrire les relations d'interaction entre les entités, et des règles et des situations sont définies pour la représentation des interactions

des entités au niveau des regroupements des entités. De plus, des règles et des situations sont définies pour déterminer dans quelle situation les relations d'interaction entre les pièces peuvent être représentées au niveau de la famille de produits; Les graphes des contacts sont utilisés pour représenter les relations de jonction entre les pièces au niveau de la famille de produits et celles au niveau de la variante des produits.

Chapitre 3

Modélisation de la génération de processus de fabrication reconfigurable

Contrairement au processus de fabrication classique qui est conçu pour satisfaire les exigences de fabrication d'un seul produit ou pièce, la génération de processus de fabrication reconfigurable satisfait aux exigences de fabrication de toutes les variantes de produits/pièce dans une famille de produits/ pièces. Dans cette section, le processus de fabrication reconfigurable pour une famille de produits dont les variantes sont des ensembles mécaniques est défini comme le processus d'assemblage reconfigurable (RAPP), tandis que le processus de fabrication reconfigurable pour une famille de pièces dont les variantes sont des pièces d'usinage est défini comme le processus d'usinage reconfigurable (RMPP). Les processus pour des regroupements d'entités sont définis comme un processus d'opération reconfigurable (RMOP).

3.1 Processus d'assemblage reconfigurable basé sur les graphes AND/OR

Le processus d'assemblage reconfigurable constitue les processus d'assemblage de toutes les variantes de produits dans une famille de produit. Il est défini comme suit:

Définition 22. Le processus d'assemblage reconfigurable (RAPP) se compose d'un en-

semble de processus d'assemblage modulaires, chacun pouvant satisfaire toutes les exigences d'assemblage d'un groupe de composants.

Les composants impliqués dans un processus d'assemblage modulaire sont définis comme un module d'accouplement. Les informations d'un module d'accouplement sont représentées dans un modèle connecté qui fait partie du modèle pour une famille de produits. Selon le modèle connecté d'un module d'accouplement, le processus d'assemblage pour ce module d'accouplement peut être généré. Nous utilisons un graphe AND/OR pour représenter les processus d'assemblage pour un module d'accouplement. Par conséquent, la représentation d'un processus d'assemblage reconfigurable consiste en un ensemble de graphes AND/OR pour tous les modules d'accouplement d'une famille de produits. Correspondant au modèle de variété de produit proposé au chapitre 2, le RAPP est modélisé de la manière suivante :

Un RAPP est représenté par un 2-tuple $< G_{PF}, G_{PV} >$, où

- G_{PF} est un ensemble de graphes AND/OR, dont chacun s'associe à un module d'accouplement au niveau de la famille de produits.
- G_{PV} est un ensemble de graphes AND/OR, dont chacun s'associe à un module d'accouplement au niveau de la variante de produits.

3.2 Processus d'usinage reconfigurable basé sur le graphe orienté

Un processus d'usinage reconfigurable (RMPP) est un processus pour une famille de pièces. L'objectif de RMPP est de fournir des processus d'usinage faisables pour chaque variante de pièces dans une famille de pièces. Dans la génération des processus d'usinage, on détermine d'abord les opérations d'usinage et leurs séquencement pour chaque entité d'usinage, puis on génère le processus d'usinage par séquencement des opérations d'usinage de toutes les entités sur une pièce. Parce que les entités des variantes de pièces dans une famille de pièces sont organisées en regroupements des entités, au lieu de générer les séquences d'usinage pour chaque variante d'entités, la génération de processus d'usinage reconfigurable génère les séquences d'opérations pour chaque regroupement d'entités. Les séquences des opérations pour un regroupement d'entité sont représentés dans un processus d'opération reconfigurable.

Définition 28. Un processus d'opération reconfigurable (RMOP) se compose d'un

ensemble des séquences des opérations similaires qui satisfont à toutes les exigences d'usinage de toutes les variantes d' entités dans un regroupement d'entités.

Un RMOP est représenté par un graphe orienté $G(V, E)$, où V est un ensemble de sommets et E est un ensemble des paires ordonnées de sommets appelées arcs. Il a les propriétés suivantes :

- Chaque noeud en V est une opération d'usinage sélectionnée pour fabriqué les variantes dans un regroupement d'entités;
- Un arc ordonné écrit comme (o_{p1}, o_{p2}) exprime que l'opération o_{p1} précède l'opération o_{p2};
- Il existe au moins un sommet de départ dans $G(V, E)$. Un sommet o_p est un sommet de départ s'il y a un chemin dirigé de o_p vers d'autres sommets de G, et pas de chemin dirigé vers o_p.
- Il existe au moins un sommet final dans $G(V, E)$. Un sommet o_p est un sommet final s'il ne précède aucun autre sommet;
- Un processus d'opération pour une variante d'entités spécifiques est un chemin ouvert dirigé partant d'un sommet de départ et se terminant par la capacité d'usinage satisfaisant à toutes les spécifications de conception. Dans une sequence des opérations, chaque sommet est visité au maximum une fois.

Des exigences d'usinage d'un regroupement d'entités correspondent aux spécifications de conception liées au processus de toutes les variantes des entités dans ce regroupement, dont les valeurs sont représentées par des intervalles ou des ensembles. Des capacités d'une séquence d'opération sont les capacités de la dernière opération de cette séquence, dont les valeurs peuvent également être représentées sous forme d'intervalles ou d'ensembles. Les capacités d'une séquence d'opérations sont recalculées une fois qu'une nouvelle opération d'usinage est rattachée à la séquence. Le mécanisme de génération de RMOP continue à ajouter des opérations plus précises dans la séquence jusqu'à ce que toutes les exigences d'usinage soient satisfaites par les capacités de la séquence d'opération.

Définition 30. Un processus d'usinage reconfigurable (RMPP) comparte les processus d'opération reconfigurable correspondant à tous les regroupements des entités d'une famille

de pièces et les relations de précédence entre les composantes des variétés des pièces. Il est représenté par un 2-tuple $< \psi, \Omega >$, où

- ψ est un ensemble de 2-tuples $< fc, G_{fc} >$, où
  - fc est un regroupement d'entités d'une famille de pièce, $fc \in FC$;
  - G_{fc} est un processus d'opération reconfigurable pour un regroupement d'entités fc représenté comme un graphe orienté.
- Ω est un ensemble des relations de précédence entre les composantes des variétés des pièce (des regroupements d'entités ou des variantes d'entités)

3.3 Conclusions

Ce chapitre se concentre sur les modèles pour la représentation d'un processus de fabrication reconfigurable. Un processus de fabrication reconfigurable est défini comme un ensemble d'éléments de processus modulaires, chacun satisfaisant à toutes les exigences de fabrication d'un groupe de composants de variétés. En réponse aux deux niveaux de variétés dans un système de produit mécanique: variétés des produits et variétés des pièces, deux niveaux de processus de fabrication reconfigurable ont été étudiés: processus d'assemblage reconfigurable pour une famille de produits et processus d'usinage reconfigurable pour une famille de pièces. Un modèle de graphe AND/OR est défini pour représenter les séquences d'assemblage faisables pour un module d'accouplement dans une famille de produits, alors un processus d'assemblage reconfigurable se compose d'un ensemble de graphes AND/OR, chacun associé à un module d'accouplement à soit au niveau de la famille des produits, soit au niveau de la variante du produit. Pour la modélisation de processus d'usinage reconfigurable, un processus d'opération reconfigurable est défini pour les séquences d'opérations faisables pour les graphes d'entités d'une famille de pièces, et un graphe orienté est utilisé pour représenter un RMOP; un processus d'usinage reconfigurable se compose d'un ensemble de processus d'opération reconfigurable en combinaison avec les relations de précédence entre les composants des variétés des pièces.

Chapitre 4

Génération de processus de fabrication reconfigurable

Ce chapitre traite des approches pour la génération d'un processus de fabrication reconfigurable. Il répond à la question : « comment traiter les informations de conception représentées par les modèles proposés au chapitre 2 afin de générer les processus reconfigurables représentées par les modèles proposés au chapitre 3 ». Puisque le concept de processus reconfigurable est défini à deux niveaux: RAPP pour une famille de produits et RMPP pour une famille de pièces, ce chapitre est consacré à proposer des approches et des algorithmes pour la génération de RAPP ainsi que pour la génération de RMPP.

4.1 Génération de processus d'assemblage reconfigurable

Puisque un RAPP est composé des processus d'assemblage modulaires, la génération de RAPP peut être décomposée par la génération de chaque processus d'assemblage modulaire. De plus, les informations relatives aux processus d'assemblage sont représentées par des modèles au niveau de la famille de produits et au niveau de la variante de produits. La génération de RAPP devrait donc envisager la génération de processus d'assemblage modulaire au niveau de la famille de produits ainsi qu'au niveau de la variante de produits.

Pour générer des processus d'assemblage modulaire, une approche de désassemblage est proposé. Le principe de l'approche de désassemblage consiste à décomposer d'abord un assemblage en un ensemble de sous-ensembles éligibles, puis à appliquer de manière récursive le processus de décomposition aux sous-ensembles générés par la décomposition précieuse

jusqu'à ce qu'aucune décomposition supplémentaire ne puisse être appliquée. Le processus inverse de chaque décomposition des sous-ensembles à leur assemblage direct forme une tâche d'assemblage faisable dans un processus d'assemblage et la séquence inverse entre toutes les décompositions de l'état terminé (les pièces de l'assemblage) à l'état initial (l'assemblage) est une séquence d'assemblage à l'intérieur d'un processus d'assemblage faisable.

L'approche de désassemblage proposée repose sur le modèle d'une famille de produits qui représente les informations d'accouplement entre les composants de variantes de produits dans la famille. L'approche comprend une procédure de pré-processus et une procédure de génération de toutes les branches faisable d'un graphe. Un algorithme général est proposé pour réaliser l'approche de désassemblage.

La procédure de pré-processus est conçue pour réduire le nombre de sommets pendant l'analyse du modèle. Il se compose d'une série des processus de test, de marquage et d'un processus de génération de graphes réduits. Dans ces processus, quatre décisions sont prises pour déterminer les attributs clés de chaque pièce représentée par chaque sommet dans le modèle. Ces attributs clés permettront à la procédure suivante de générer les ensembles de branches faisable.

Le procédure clé de l'approche de désassemblage est la génération de toutes les coupes faisable. Dans ce but, un algorithme de Karger amélioré est proposé. L'algorithme adopte le principe de la contraction d'arc de l'algorithme de Karger mais les arcs sont choisies par des règles prédéfinies. En utilisant ces règles, l'espace de solution peut être considérablement diminué car ces règles peuvent réduire le nombre de sommets et de arcs du graphe analysé. Des mécanismes de « backtraking » sont déployés pour guider l'algorithme pour explorer l'espace de solution entier et trouver toutes les branches faisables.

4.2 Génération de processus d'usinage reconfigurable

Basé sur la définition de RMPP, la génération de processus reconfigurables d'usinage est divisé en deux étapes : 1) la première étape consiste à générer le RMOP pour chaque cluster d'entités; 2) la deuxième étape consiste à générer les relations de précédence entre les composantes de variétés de pièce. Parce que beaucoup d'approches dans la littérature peuvent être utilisées pour générer les relations de précédence pour la deuxième étape, dans ce travail de thèse, on suppose que les relations de précédence entre les variantes d'entités

sont déjà données. Les relations de précédence peuvent être représentées à la fois au niveau du regroupement d'entités et au niveau de variante d'entités selon les règles définies au chapitre 3. Les relations de précédence données sont ensuite utilisées pour configurer les processus d'usinage pour une variante de pièces spécifique par l'approache proposée dans Chapitre 5. Ce travail se concentre uniquement sur l'étape 1 qui est la génération de RMOP pour un cluster d' entités.

Afin de générer le RMOP pour un regroupement d'entités, une approche en deux étapes est proposée. Le principe de l'approche proposée est de faire correspondre les capacités d'usinage des opérations avec les exigences d'usinage correspondantes d'un regroupement d'entités. Le processus d'appariement est divisé en deux étapes:

- Dans la première étape, toutes les opérations d'usinage faisables sont d'abord choisies parmi un ensemble de opérations d'usinage selon leurs compatibilités géométriques par rapport aux spécifications géométriques du regroupement d'entités. La logique de premier ordre est utilisée pour représenter la connaissance des capacités géométriques des opérations d'usinage. Une base de connaissances est conçue pour structurer la représentation de ces connaissances de sélection des opérations. De plus, un algorithme de résolution basé sur R-BF est proposé pour sélectionner toutes les opérations géométriquement compatibles avec un cluster d' entités.

- Dans la seconde étape, ces opérations d'usinage faisables sont séquencées selon leurs capacités de tolérances et de leurs capacités de finition de surface pour satisfaire les spécifications correspondantes au cluster d'entités. Un modèle mathématique est proposé pour modéliser le problème de séquençement des opérations et un algorithme de parcours en profondeur est proposé pour résoudre ce problème.

4.3 Conclusions

Ce chapitre traite de la génération de RPP à partir de deux niveaux de granularité. Au niveau de la famille de produits, la génération d'un RAPP est décomposée en générations d'un ensemble de processus d'assemblage modulaire pour les composants des variétés de produits. Une approche de désassemblage basée sur le découpage de graphe est proposée pour générer le processus d'assemblage modulaire en analysant les informations liées au processus représentées par le modèle DFC. Le resultat de cette approche est un graphe

AND/OR représentant le processus d'assemblage modulaires. Les graphes AND/OR de tous les processus d'assemblage modulaires d'une famille de produits composent le RAPP final.

Au niveau de la famille de pièces, la génération de RMPP est divisée en deux étapes: la génération des RMOPs et la génération des relations de précédence entre les composantes de variétés. Puisque de nombreuses approches sont disponibles dans la littérature pour la génération de les relations de précédence, on se concentre uniquement sur l'approche pour générer des RMOPs. Dans ce but, une approche en deux étapes est proposée: toutes les opérations d'usinage faisables sont sélectionnées dans la première étape, puis dans la deuxième étape, les opérations sélectionnées sont séquencées pour générer le RMOP pour un groupe de fonctions.

Les modèles de représentation proposées au chapitre 2 sont utilisés pour fournir les informations liées au processus pour la génération de RPP. Les sorties des algorithmes sont conformes aux modèles de RPP proposés au chapitre 3. Une famille de pompes à engrenages et une famille de pompes à huile sont utilisés pour montrer la faisabilité et l'efficacité des approches proposées. Les processus de fabrication pour un produit/pièce spécifique dans la famille sont dérivés en configurant le RPP existant. Dans le chapitre suivant, on propose des approches pour intégrer le RPP avec la configuration produit/pièce et un système de fabrication afin de générer le processus de fabrication complet et optimal pour un variante de produits/pièces.

Chapitre 5

Sélection et optimisation du processus pour une variante spécifique

Le model RPP contraint l'ensemble des processus de fabrication pour toutes les variantes de produits/pièce sur les aspects des opérations du processus et leur séquencement. Ces points communs peuvent être identifiés dans le RPP avant qu'une variante de produit/pièce spécifique ne soit attribuée à un système de production. De cette façon, le RPP augmente l'efficacité de la génération de processus de fabrication pour les variantes de produits/pièces susceptibles de changer dans un environnement de fabrication dynamique. Ce chapitre est consacré à l'application de ces concepts de RPP. Deux phases d'application ont été identifiées pour montrer des capacités du RPP dans un système de fabrication dynamique et variable.

La phase I est dédiée à l'intégration de la configuration du produit/pièce et de la configuration du RPP. Dans cette phase, les processus de fabrication modulaires et les relations de priorité correspondantes pour une variante de produits et de pièces spécifiques sont sélectionnés pendant le processus de configuration de cette variante de produits/pièces. Ils correspondant avec les éléments constitutifs des processus de fabrication finaux pour une variante de produits/pièces et sont utilisés dans la phase II pour élaborer le processus de fabrication optimal.

Dans la phase II, le processus final d'une variante de produit/pièce spécifique est construit à partir des éléments de processus pour atteindre les objectifs d'optimisation dans un scénario de fabrication donné. Deux scénarii de fabrication sont identifiés pour l'introduction d'un nouveau produit/pièce, et l'objectifs d'optimisation et les entités objectives permettant

d'évaluer les processus candidats sont définis pour ces deux scénarii. Les algorithmes sont ensuite proposés pour rechercher la solution optimale.

Ce chapitre traite des deux étapes d'application du RPP à partir de deux niveaux de granularité: le processus d'assemblage reconfigurable (RAPP) pour une famille de produits et le processus d'usinage reconfigurable (RMPP) pour une famille de pièces. Bien que les implémentations pour ces deux niveaux de granularité soient différentes, les principes de base pour leurs applications sont similaires.

5.1 Integration de la configuration du produit/pièce et de la configuration du RPP

Une application du RPP proposé est l'intégration de la configuration du produit/pièce et de la configuration du RPP. Dans cette application, les éléments du processus liés à une variante de produits/pièces peuvent être immédiatement dérivés du RPP lorsque la variante est configurée. Étant donné qu'un RPP contient les éléments de processus pour toutes les variantes d'une famille, lorsqu'une variante est dérivée, seuls les éléments de processus liés à cette variante doivent être sélectionnés pour générer les processus de fabrication de cette variante. En termes de RAPP, les éléments de processus pour une variante de produit sont les processus d'assemblage modulaires pour les modules d'accouplement qui ont des composants impliqués dans cette variante de produits; en termes de RMPP, les éléments de processus pour une variante de pièces incluent les ROPPs pour ses regroupements d'entités et les relations de précédence liant les variantes fonctionelles en interaction sur cette variante de pièces.

Le problème de l'intégration de la configuration du produit/pièce et de la configuration du RPP est modélisé comme un problème de satisfaction de contraintes dynamiques (DCSP). Dans un DCSP, chaque variable peut prendre un état: soit actif soit inactif.Ainsi, seules les variables actives sont impliquées dans le processus de distribution de valeur. Outre les deux états des variables, des contraintes d'activité sont introduites pour spécifier les conditions dans lesquelles les variables deviennent actives. Le processus de résolution commence par un ensemble de variables actives initiales et les affectations de ces variables. D'autres variables sont activées dans le processus de résolution dès que les contraintes d'activité impliquant ces variables sont satisfaites. De façon correspondante, une contrainte est "active" si toutes les

variables liée à cette contrainte sont actives. Sinon, elle est "inactive". Seules les contraintes actives sont propagées dans le processus de résolution du problème. Grâce à ce mécanisme dynamique, l'efficacité de la recherche de solutions peut être améliorée. Puisque RPP a deux niveaux de granularité : RAPP et RMPP, la représentation par DCSP est proposée à chaque niveau de ce problème.

L'intégration de la configuration du produit et de la configuration du RAPP sont modélisées comme un problème de DCSP sous la forme d'un tripler $I =< V, D, C >$ où:

- $V = \{V_{\mathrm{a}}, V_{\mathrm{f}}, V_{\mathrm{sa}}, V_{\mathrm{pf}}, V_{\mathrm{pv}}, V_{\mathrm{ma}}\}$ est un ensemble de variables, divisé en six sous-ensembles. V_{a} est un ensemble de variables constitué de toutes les variables d'attributs configurables. V_{f} est un ensemble de variables dans lequel chaque variable correspond à une composant functionalle défini dans l'architecture de décomposition d'une famille de produits. V_{sa} est un ensemble de variables dans lequel chaque variable correspond à un sous-assemblage défini dans l'architecture de décomposition. V_{pf} est un ensemble de variables dans lequel chaque variable correspond à une famille de pièces définie dans l'architecture de décomposition. V_{pv} est un ensemble de variables dans lequel chaque variable correspond à une variante de pièce définie dans dans l'architecture de décomposition. V_{ma} est un ensemble de variables dans lequel chaque variable correspond à un processus d'assemblage modulaire dans le RAPP d'une famille de produits.

- $D = \{D_{\mathrm{a}}, D_{\mathrm{b}}\}$ est l'ensemble des domaines des variables de V. D_{a} est l'ensemble des domaines qui contientnent toutes les valeurs d'attributs pour une variable de V_{a}. D_{b} est un domaine booléen : $\{0, 1\}$ pour toutes les variables dans V_{f}, V_{sa}, V_{pf}, V_{pv} et V_{ma}. Si les variables dans V_{f}, V_{sa}, V_{pf}, V_{pv} et V_{ma} prennent la valeur 0 de D_{b}, alors les composants configurables correspondants à ces variables ne sont pas choisis dans la configuration finale, sinon les variables prennent la valeur 1.

- $C = \{C_{\mathrm{c}}, C_{\mathrm{a}}\}$ est un ensemble de contraintes dont chacune limite les valeurs des variables de V. C est divisé en deux sous-ensembles, C_{c} et C_{a}. C_{c} est un ensemble de contraintes de compatibilité sur les variables de V et C_{a} est un ensemble de contraintes d'activité. Des contraintes de compatibilité C_{c} définissent les relations de sélectivité entre les composants configurables d'une famille de produits. Les contraintes d'activité C_{a} décrivent des conditions dans lesquelles une variable peut ou non être activement considérée comme faisant partie d'une solution finale. Des contraintes de compatibilité

et des contraintes d'activité sont exprimées par le schéma de représentation basé sur la logique propositionnelle proposé au chapitre 2.

De même, l'intégration de la configuration d'une pièce et de la configuration du RMPP est aussi modélisé comme un problème de DCSP au triple $I =< V, D, C >$ où:(Xia et al., 2016):

- V=$\{V_{\mathrm{f}}, V_{\mathrm{fc}}, V_{\mathrm{fv}}, V_{\mathrm{rm}}, V_{\mathrm{pr}}\}$ est un ensemble de variables constitué de quatre sous-ensembles, V_{f}, V_{fc}, V_{fv}, V_{rm} et V_{pr}. V_{f} est un ensemble de variables correspondant aux fonctions techniques au niveau des modules fonctionnels dans le modèle de variété de pièces. V_{fc} est un ensemble de variables correspondant aux regroupements d'entités dans le modèle de variété de pièces. V_{fv} est un ensemble de variables correspondant aux variantes d'entités dans le modèle de variété de pièces. V_{rm} consiste en un ensemble de variables correspondant aux RMOPss dans le RMPP. V_{pr} comprend un ensemble de variables correspondant aux relations de précédence entre des composantes de physique d'une famille de pièces. Toutes les variables dans V sont Boolean variables.
- $D = \{0, 1\}$ est un domaine booléen pour toutes les variables dans V.
- $C = \{C_{\mathrm{c}}, C_{\mathrm{a}}\}$ est un ensemble de contraintes C_{c} est un ensemble de contraintes de compatibilité et C_{a} est un ensemble de contraintes d'activité.

Deux exemples illustratifs sont étudiés pour démontrer l'application de l'intégration de la configuration d'un produit ou pièce et de la configuration du RPP. La famille de pompes à engrenages et la famille de coups de pompes à huile mentionnées au chapitre 2 sont utilisées comme exemples illustratifs. L'approche basée sur le DCSP est implémentée en utilisant un système de programmation logique contrainte (CLP) : ECLiPSe.

5.2 Optimisation du processus pour une variante spécifique

Dans un environnement de fabrication dynamique, la situation de fabrication a une probabilité et une fréquence élevées de subir des changements dues à l'introduction de nouveaux produits, la panne d'une machine ou l'ajustement du volume de production. Par conséquent, le processus de fabrication dans un système de fabrication peut également nécessiter d'être modifié dans des situations de fabrication différentes. Donc, la génération de processus de fabrication doit être suffisamment flexible pour répondre à ces besoins d'agilits.

Étant donné que le RPP intègre tous les processus de fabrication réalisables pour toute variante de produit/pièce dans une famille donnée, les éléments de processus modulaires dérivés de l'intégration de la configuration du produit/pièce et de la configuration du RPP contiennent toutes les informations nécessaires pour construire les processus réalisables pour une variante de produit/pièce spécifique. Lorsqu'il est nécessaire d'adapter une nouvelle dynamique de fabrication, on peut construire un processus optimal à partir des éléments du processus pour cette nouvelle situation de fabrication sans pour autant la générer à partir de zéro. De cette façon, le RPP fournit une méthode flexible de génération de processus.

Après la phase I d'application, dans la phase II, la situation du système de fabrication actuel est prise en compte et le processus final et optimal pour ce système de fabrication est généré pour atteindre les objectifs d'optimisation.

Selon la dynamique rencontrée par un système de fabrication, différents objectifs d'optimisation et conditions de contraintes doivent être définis. Dans cette thèse, deux scénarios de fabrication sont considérées : (Xia et al., 2016)

- Scénario 1 : La production passe d'une variante de produit/pièce à une autre dans un système de fabrication flexible avec une production à faible volume. Les objectifs d'optimisation pour une sélection optimale du processus dans ce scénario sont : 1) minimiser le délai de production; et 2) sélectionner le processus pour les variantes actuelles aussi proche que possible de celui de la variante précédente.

- Scénario 2 : La production passe d'une variante de produit/pièce à une autre variante de produit/pièce dans un système de fabrication reconfigurable avec une production à grand volume. Le scénario 2 étant différent du scénario 1, le système de fabrication est reconfigurable et le volume de production de la nouvelle variante introduite est grand. Les objectifs d'optimisation pour une sélection optimale du processus dans ce scénario sont : 1) minimiser le coût du processus; 2) minimiser le coût de la ressource de fabrication et de la configuration du système.

5.2.1 Optimisation du processus d'assemblage pour une nouvelle variante de produit

Au niveau du produit, cette étape concerne la construction et l'optimisation des processus d'assemblage pour une nouvelle variante de produit dans un système de fabrication spéci-

fique. En appliquant l'intégration de la configuration de produit et de la configuration de RAPP, un ensemble de graphs modulaires AND/OR sont sélectionnés à partir du RAPP au moment où un ensemble de variantes de pièce pour la nouvelle variante de produit est sélectionnée. Ces graphes modulaires sont ensuite assemblés pour former un graphe complet AND/OR pour cette variante de produits qui représente toutes les séquences d'assemblage réalisables.

Un algorithme pour la construction d'un graphe complet AND/OR à partir d'un ensemble de graphes AND/OR modulaires est proposé. Un graphe complet AND/OR contient tous les séquence d'assemblage faisables pour une variante de produit. Ainsi, après avoir générer le graphe complet, le séquence d'assemblage optimal peut être trouvé en appliquant une approche de recherche sur ce graphe.

Un algorithme AO* est appliqué pour rechercher la séquence d'assemblage optimale à partir du graphe complet AND/OR d'une variante de produit. Compte tenu des différents objectifs d'optimisation dans les deux scénariis de fabrication mentionnés précédemment, différentes méthodes de calcul de l'estimation heuristique des nœuds et du coût des hyperarcs dans le graphe AND/OR sont proposées pour cet algorithme.

5.2.2 Optimisation du processus d'usinage pour une nouvelle variante de produit

Après la phase I, tous les éléments de processus pour une nouvelle variante de pièces sont dérivés de l'integration de la configuration d'une pièce et de la configuration du RMPP, y compris toutes les entités d'usinage de cette variante de pièces et les RMOPs liées à ces entités ainsi que les relations de précédence entre ces entités.

Afin de trouver la séquence d'usinage optimale pour une nouvelle variante de pièce, l'application dans la phase II aux niveau du pièce est divisée en quatre étapes :

- Étape 1 : Trouver la séquence d'opération d'usinage optimal pour chaque entité. Un algorithme de recherche « best-first » est développé pour cette étape. Les méthodes de calcul du coût des arcs et l'estimation du coût des noeuds dans le graphe orienté d'un RMOP sont proposées.

- Étape 2 : Établir les contraintes de priorité entre les opérations d'usinage. Les contraintes de priorité proviennent de deux aspects: 1) des relations de précédence

d'entités; et 2) des séquences d'opérations d'usinage pour les entités d'une pièce. Des règles pour déterminer les contraintes d'précédence sont proposées.

- Étape 3 : Élaborer les objectifs et les entités d'optimisation. Dans cette étape, les objectifs d'optimisation et les entités pour évaluer des solutions pour une séquence d'opération optimale sont proposé.

- Étape 4 : Appliquer un algorithme d'optimisation pour trouver la séquence d'opération optimale. L'étape finale consiste àutiliser un algorithme d'optimisation pour trouver la séquence d'opération optimale qui minimise la fonction objectif définie à l'étape 3, tout en satisfaisant toutes les contraintes de priorité définies dans l'étape 2. De nombreuses approches d'optimisation sont disponibles dans la littérature, par conséquent, cette étape n'est pas détaillée dans cette thèse.

5.3 Conclusions du chapitre

Ce chapitre répond à la question : comment appliquer le RPP pour gérer la complexité de fabrication induite par la variété de produits/pièces. L'application du RPP se décompose en deux phases. Dans la première phase, une variante de produit/pièce et tous les éléments de processus sont générés par l'intégration de la configuration du produit/pièce et de la configuration du RPP. Une approche basée sur le DSCP est développée pour réaliser l'intégration de la configuration du produit/pièce et de la configuration du RPP.

Dans la deuxième phase, des éléments de processus dérivés sont utilisés pour établir la séquence d'opération optimale pour une variante de produits/pièces. La séquence d'opération optimale est adaptée à un système de fabrication spécifique en tenant compte des ressources de fabrication disponibles et des objectifs d'optimisation correspondants. Deux scénariis de fabrication sont identifiés pour la phase II. Chaque scénario de fabrication a ses propres objectifs d'optimisation. Les approches d'optimisation du processus d'assemblage et d'optimisation du processus d'usinage dans les deux scénarios sont étudiées respectivement.

Les approches et les algorithmes proposés peuvent être mis en œuvre dans des programmes informatiques pour automatiser les deux phases d'application et chaque proposition est suivie d'un exemple pour illustrer sa faisabilité.

Conclusions, limites et perspectives

Conclusions

Dans les paradigmes de fabrication actuels, les approches conventionnelles de la génération de processus de fabrication sont inefficace pour gérer la complexité induite par la variété de produits et l'agilité attendue des systèmes de fabrication, car ils traitent chaque variante de produits/pièces individuellement. Le RPP est une nouvelle approche qui vise à générer le processus pour une famille de produits/pièces.

La recherche présentée dans cette thèse apporte une contribution majeure à la méthodologie, à l'architecture, aux modèles de représentation et aux algorithmes de génération de processus reconfigurables à deux niveaux de granularité: une famille de produits et une famille de pièces. Dans cette thèse, une famille de produits/pièce est définie comme un domaine de variantes où ensembles et intervalles sont utilisés pour représenter les valeurs d'attribut de leurs caractéristiques de conception. De cette façon, le nombre de variantes dans une famille de produits ou parties peut être potentiellement infini.

Deux types de modèles de représentation sont définis dans cette thèse. Le modèle de variétés de produits basé sur le concept d'entité représente les spécifications de conception d'une famille de produits/pièces d'une manière structurée et modulaire. Il fournit les données nécessaires pour générer des RPP ainsi que les informations pour configurer une variante de produits/pièces. Les modèles de RPP, y compris RAPP et RMPP, fournissent des méthodes basées sur des graphes pour représenter des processus modulaires pour une famille de produits/pièces donnée.

Les approches et les algorithmes sont proposés pour générer les processus de fabrication reconfigurables. La génération de processus reconfigurables est étudiée à partir de deux niveaux de granularité : la génération de RAPP et la génération de RMPP. Les modèles de

variétés de produits proposées sont utilisés pour fournir les informations liées au processus de génération de RPP. Les resultats des algorithmes sont conformes aux modèles de RPP proposés.

Afin de répondre à la question: « comment appliquer le RPP pour gérer la complexité de fabrication induite par la variété de produits/pièces », deux phases RPP sont proposées. Dans la première étape, une variante de produit/pièce et tous les éléments de processus sont générés par l'intégration de la configuration du produit/pièce et de la configuration du RPP. Une approche basée sur le DSCP est développée pour réaliser l'intégration de la configuration du produit/pièce et de la configuration du RPP.

Afin d'illustrer les modèles de représentation proposés, une famille de pompes à engrenages et une famille de coups de pompes à huile sont utilisées. Les informations de ces deux exemples sont détaillées en annexe. Pour tous les algorithmes proposés dans cette thèse, des exemples sont suivis pour montrer leurs résultats et faisabilité.

Comparativement aux approches conventionnelles, les approches RPP proposées dans cette thèse présentent deux principaux avantages :

- Réduire la complexité de la génération de processus pour une famille de produits/pièces;
- Améliorer la flexibilité de la génération de processus pour suivre les besoins dynamiques du système de fabrication.

Limitations

Les approches proposées dans cette thèse ont cependant plusieurs limites suivantes :

- Les approches proposées dans cette thèse ne s'appliquent qu'aux variantes de produits/pièce dont les spécifications de conception sont contenues dans le modèle de variété de produits/pièces.
- Le processus d'assemblage considéré dans cette thèse est limité à un processus séquentiel, non linéaire et monotone.
- Les approches de génération de processus proposées se limitent à la génération conceptuelle de processus dans laquelle la sélection des processus et leur séquencement sont leurs principaux axes.

- Dans l'approche pour l'integration de la configuration du produit/pièce et de la configuration de RPP, seuls les domaines finis et les domaines booléens sont pris en compte pour les variables de configuration.
- Dans l'etat actuel des travaux, les contraintes de configuration pour l'integration de la configuration du produit/pièce et de la configuration de RPP sont générées manuellement.
- Compte tenu des approches d'optimisation existantes dans la littérature, les implémentations pour l'optimisation du processus ne sont pas détaillées dans cette thèse.

Perspectives

Pour éclairer les travaux futurs, les questions suivantes doivent examinées :

- Développement de l'estimation des coûts de l'algorithme AO * pour le rendre plus proche du coût réel du noeud dans le graphe de recherche en termes d'objectifs d'optimisation.
- Augmentation du nombre de scénariis d'application pour le RPP. Un travail futur intéressant pourrait être l'intégration de la configuration du système de fabrication et de la configuration du RPP.
- Validation dans un cas industrielle.
- Automatisation, dans une certaine mesure, de la génération de contraintes de configuration pour l'intégration de la configuration du produit/pièce et de la configuration du RPP.

Part II

English Version

Introduction

Motivation

In order to survive in today's fierce global competitions, manufacturers are striving to provide customized products to satisfy varying customer demands. As a result, the number of product variants offered in most industries has increased dramatically over the past few decades. Although product variety can boost the competitive power of companies, expand their market and promote their sale volume, these positive outcomes are not always guaranteed unless variety is well-managed during all the stages of product life-cycle (ElMaraghy et al., 2013). One challenge for manufacturers today is to achieve the economies of scope without losing too much on the economies of scale.

The economic, socio-political and technological dynamics have great influences on the manufacturing activities of today's manufacturers. These manufacturing dynamics require the product system, process system and production system of a company to respond concurrently and cooperatively. For example, the high turbulence of market demand causes a short product life cycle, consequently the manufacturing process plans and manufacturing system need to adapt in time to the constant introduction of new products. Therefore, how to agilely respond to the manufacturing dynamics while delivering product variety has become another great challenge faced by manufacturers.

Two manufacturing paradigms have been proposed by the academy and the industry to cope with the challenges. One manufacturing paradigm is Mass Customization (MC), which aims to deliver affordable product variety with near mass production efficiency without compromising cost, quality or delivery time (Pine, 1993). Another one is co-evolution of Product, Process and Production System (P3S). In this paradigm, the manufacturing dynamics, namely "changes", are handled by repetitive configuration of product, process

and production system over time (Tolio et al., 2010).

Product family design is an effective strategy for the two manufacturing paradigms as it allows product variants to share a number of common components and functions while each product variant can have its unique specifications to meet customer's demands (Jiao et al., 2007). Organizing product variants into product family can not only decrease the design complexity for product variety, but also ease the production of different product variants.

Process planning is a knowledge intensive and complex task that transforms design information into manufacturing processes and determines the optimal sequence of operations. As a connecting bridge between product design and production, manufacturing process planning plays a key role in maintaining high levels of responsiveness and adaptability for manufacturing system while propagating variety from product to process. The benefits from product family design cannot be guaranteed if there is no appropriate process planning methods transforming efficiently product variants into manufacturing process variants which realize them.

Conventional process planning methods like variant process planning based on group technology, generative process planning based on artificial intelligence, non-linear process planning and adaptive process planning, mainly focus on the process plans generation for one individual product/part. These either pre-planning or re-planning methods have obvious disadvantages under the circumstance of product family:

- They seldom consider the future changes of the product family model and the available production resource;
- The generated process plans are prone to be obsolete for new product/part variants;
- Large computational burden to regenerate the process plans for new product/part variants.

Being aware of the weaknesses of traditional process planning methods on the support to the changeable and responsive manufacturing paradigm, ElMaraghy (2007) proposes a new concept of process planning, that is "Reconfigurable process planning (RPP)", to enable the changeability of process plans for evolving products and manufacturing systems. One method for RPP is proposed by Azab et al. (2007), which firstly identify the new and missing features/operations by comparing the new product/part with original ones, and then to

adapt the new changes by configuring the existing process plans instead of generating a new one from scratch.

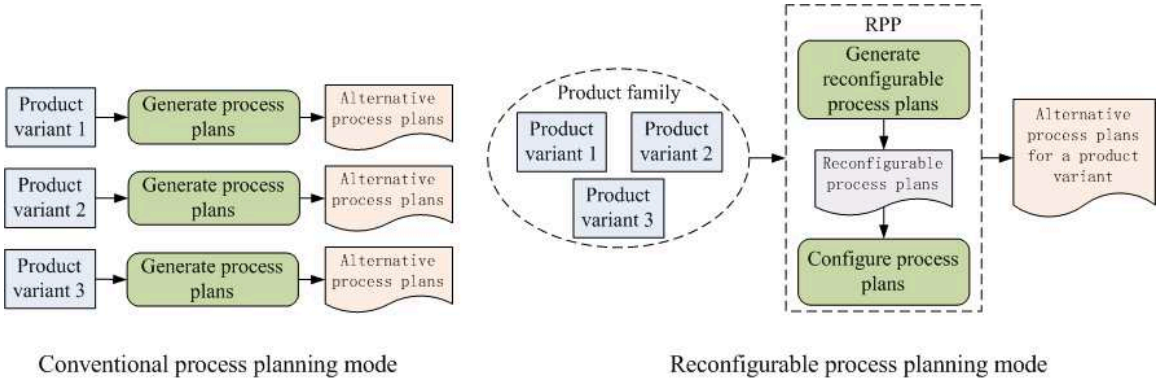


Figure 5-1: Comparison between RPP and conventional planning method for product variety

RPP adopts configuration technology in the activity of process planning for product variety. The process variants for new product variants are generated by configuring the existing process components with the consideration of the available manufacturing resources. RPP is superior to the traditional process planning methods in the aspects of efficiency and changeability, and it has been identified as one of the key enablers for the realization of MC and co-evolution of P3S (ElMaraghy et al., 2013; Tolio et al., 2010). However, as an emerging technology, RPP is still far away from industrial application. Before being applied into the industry, RPP should exhibit its capabilities to cope with the following scenarios:

- Scenario I - Handling the variation at different granularity levels of product design
  The design variation for a product could frequently occurs at product level or part level or feature level, for example, a portion of parts in a product are changed, or a portion of features on a part are modified. No matter where the variation takes place, RPP should be able to rapidly generate the feasible manufacturing process plans for it.

- Scenario II - Handling the variation in a dynamic manufacturing system
  The capability and availability of manufacturing resource in a manufacturing system are subject to change in a dynamic environment. RPP should have the capability to flexibly generate the optimal manufacturing process plans for a given production system and also for a changeable and reconfigurable manufacturing system.

Research Objectives

For realizing the two application scenarios of RPP, this research intends to develop a methodology, architecture, proper models and algorithms for RPP, by means of which the process plans for any product/part variant in a product/part family can be flexibly and automatically generated. It is devoted to address the following issues:

- *What is the concept of RPP?* As a new process planning method for new manufacturing paradigms, RPP is build upon the conventional process planning methods, but takes the design specifications of a product family as input instead of only the information of one single product variant. In terms of the concept of process plan, there exist six granularities considering on two dimensions as listed in figure 5-2. This thesis dedicates to develop the definitions of RPP at conceptual level.

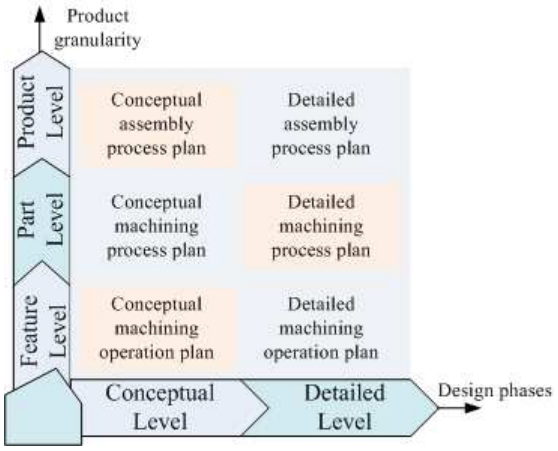


Figure 5-2: Granularities of manufacturing process plan

- *How to represent the heterogeneous information and knowledge necessary for generating and handling reconfigurable process plans?* Large amounts of product design data need to be modeled for computer-aided process planning. The input information for RPP is a product family, not a single product variant. The design specification of a product family is an aggregate of the design data of every product variant. Therefore, proper and effective structured representation for the design specification of a product family is of importance for generating reconfigurable process plans. In addition, in order to generate reconfigurable process plans automatically, the knowledge on mapping the design specification to process plan also need to be represented in a computer

interpretable way.

- *How to generate reconfigurable process plans for a product/part family?* After representing the design information and knowledge, the methods and algorithms for RPP need to be developed. These methods and algorithms use the pre-defined information and knowledge to generate reconfigurable process plans for a product family. As the concept of RPP is defined at varying levels of granularity, the methods and algorithms for RPP need to be designed with regard to each level of granularity.

- *How to apply RPP to handle the manufacturing complexity induced by product variety under the manufacturing paradigms of MC and co-evolution of P3S?* After the models and methods for RPP are developed, it is necessary to illustrate that how RPP can respond to the evolution of product and production system. The effectiveness and advantages of RPP comparing to the conventional process planning methods should also be verified.

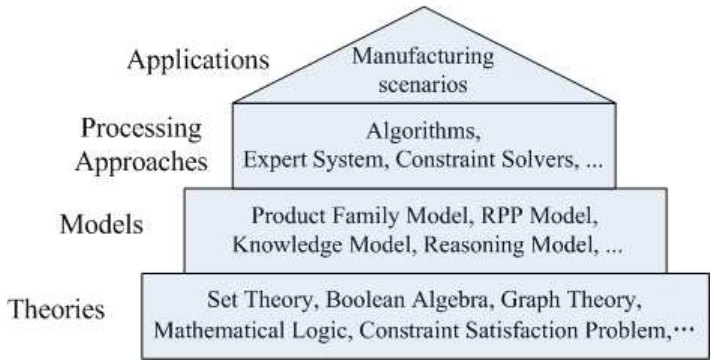


Figure 5-3: The methodology framework of this research

The methodology framework of this research is shown in figure 5-3. It consists of four levels of abstractions, from bottom to top: theories, models, processing approaches, applications. This research lays itself on a series of basic theories, for instance, set theory, Boolean algebra, graph theory, which supports the contributions of this research. Based on the theories, a set of models are developed to represent the related data, information and knowledge both on product side and process planning side. After the necessary models are ready, the approaches, tools and algorithms are designed to process and operate these models for the desired results. In the end, the application scenarios are used to test and validate the proposed approaches and models so as to prove the feasibility and effectiveness

of this research.

Organization of the thesis

Focusing on the research objectives, this thesis is organized into five chapters. After this introduction, the first chapter starts with the definitions of the basic concepts on the proposed subject to better define the research scope of the thesis. In the first chapter, extended definition for the concepts of product family/variety and part family/variety are proposed. Then it reviews the state-of-the-art on the related research including product modeling for manufacturing, configuration technology for product variety and process variety and manufacturing process planning. The main contributions of this thesis are presented in chapter 2, 3, 4, and 5.

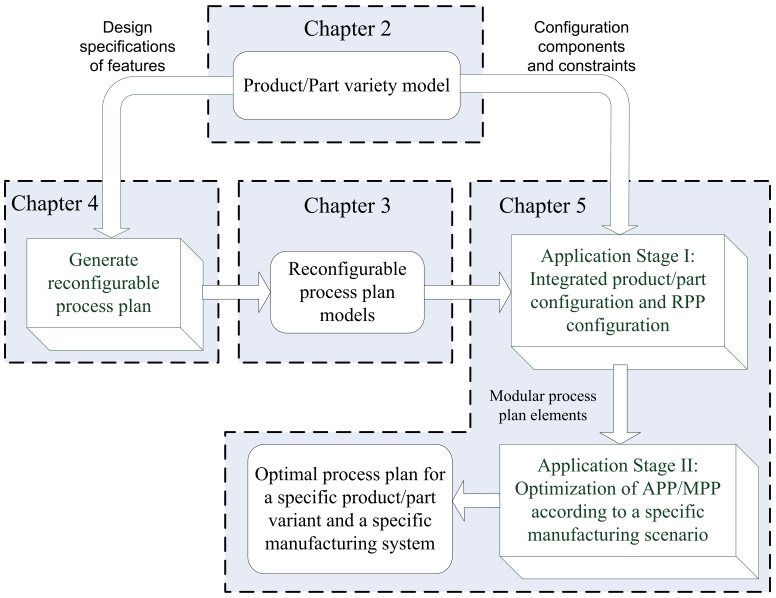


Figure 5-4: The framework of this thesis

Based on modular, platform-based and configuration-based techniques, Chapter 2 proposes structured modeling methods to represent the design specification of a product family for RPP. The aims are to elaborate what information about product families is required for RPP, and how the information can be represented systematically for a computational implementation.

Chapter 3 proposes the models to represent reconfigurable process plan. In this chapter, the concept of reconfigurable process plan are developed into two levels of granularity:

reconfigurable assembly process plan, reconfigurable machining process plan.

Chapter 4 deals with the knowledge modeling and methods for the generation of reconfigurable process plan. It answers the question that how to process the design information represented by the product variety model proposed in chapter two in order to generate the reconfigurable process plans in the representation way proposed in chapter three. First order logic is used in this chapter to model the knowledge involved in RPP. The algorithms for the generations of reconfigurable assembly process plan, reconfigurable machining process plan and reconfigurable operation process plan are developed respectively in this chapter.

Chapter 5 is devoted to the applications of RPP under the manufacturing paradigms of MC and co-evolution of P3S. A constraint based method for integrated product configuration and RPP configuration is proposed to respond to the changes in product system. Optimization objectives and evaluation methods for searching the optimal process plans for a specific product/part variant in a specific manufacturing scenario are studied. Illustrative examples are shown in this chapter to show the effectiveness and feasibility of the proposed methods.

In the conclusion, the contributions and limitations of the proposed methods in this thesis are highlighted and the issues for enlightening the future work are concluded.

Chapter 1

Definitions and related work

This chapter discusses the relevant definitions and research. The work in the domains of product modeling for manufacturing, configuration technology in product variety management and manufacturing process planning are covered. It starts with the main definitions of the related concepts from manufacturing engineering. After that, related literature is studied. The chapter ends with an evaluation of the available literature with emphasis on the necessity of this research.

1.1 Definitions of key concepts

Since the main objective of this research is to develop representation models and generation methods for RPP which is a manufacturing process planning method for a product family, the concepts of product, part, product family, part family, manufacturing process and manufacturing process planning should be clearly defined so as to show the scope of this research. The concept of product has different definitions based on different points of view. In marketing, a product is anything that can be offered to a market to satisfy a want or need (Kotler and Armstrong, 2010). In project management, a product is an artifact resulting from an organizational process (Krishnan and Ulrich, 2001). In product design, a product is a complex assembly of interacting components (Finger and Dixon, 1989). In this research, we give this concept a definition from the perspective of manufacturing:

Definition 1 (*Product*)**.** A product is a mechanical assembly produced by a sequence of assembly processes which fix gradually all components of this product together. With appropriate input, a product can function individually to fulfill certain customer's demands.

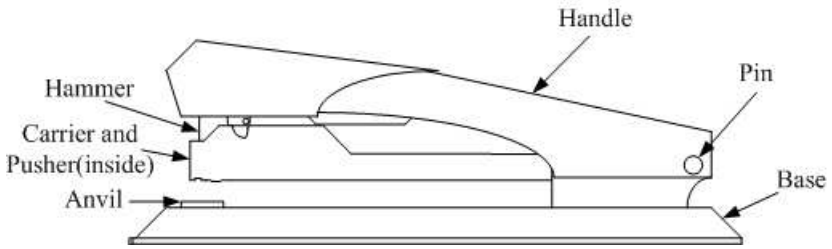


Figure 1-1: Desktop stapler

Product is a composition of a set of components in a way that the components in the product cannot fall apart under the influence of gravity or during the service process of this product. For example, a desktop stapler, as shown in figure 1-1, is a product. It is assembled from a set of components including handle, pin, base, hammer, carrier, pusher, and anvil. The assembly process for this stapler puts its components together in a way that the previous attached components cannot hinder the attachments of the following ones.

After defining the concept of product, we can define the concept of part as follow:

Definition 2 (*Part*)**.** A part is an inseparable component when assembling a product.

Part is the smallest building block for a product, and every component of a product is either a part or a composition of a set of parts. For example, the anvil of the stapler in figure 1-1 is a part, because it is only a single piece of material, while the handle of the stapler is not a part, as it is a combination of a plastic handle head and a steel handle body.

In the manufacturing paradigm of MC, product variety drives manufacturers to manage a set of products instead of one single product. In order to gain benefits from commonality and modularity, products are grouped into product families. Depending on the different context (marketing, design or manufacturing), the interpretation of product family could be different.

Meyer and Utterback (1993) define a product family as: a set of products that share a common platform but have specific features and functionality required by different sets of customers.

Simpson (1998) defines a product family as: a group of products which share common form features and function(s), targeting one or multiple market niches.

Laakko and Mäntylä (1994) define a product family in terms of process planning: a set of products which have similar routings and manufacturing processes.

Stadzisz and Henrioud (1995) consider a product family as: a set of similar products whose main functions are identical.

ElMaraghy et al. (2013) consider a product family as: a group of products based on a specific design concept, or derived from a standard parent product, and are similar in design and/or production methods.

Table 1.1: Criteria considered in different definition of production family

Definitions	Function	Feature	Design	Manufacturing process	Market
Meyer and Utterback	√	√	-	-	√
Simpson	√	√	-	-	√
Laakko and Mäntylä	-	√	-	√	-
Stadzisz and Henrioud	√	√	-	-	-
ElMaraghy, et al.	-	√	√	√	-

Some authors also use the concept of product variety to describe a set of similar product variants (Hu et al., 2011; Daaboul et al., 2011; ElMaraghy et al., 2013). The differences between product variety and product family are fuzzy in the literature. In this thesis, product variety and product family are treated as two different points of view for the same group of similar products. Product variety describes a group of similar products from a design's perspective, while product family describes them from a manufacturing point of view.

In addition, the existing concepts consider the product variety and product family as a finite set of product variants. But for an evolving product system, the number of product variants is not fixed. Thus, in this thesis, the concepts of product variety and product family are extended to be a infinite set of product variants by involving a mathematical definition of "domain". In mathematics, a domain could be a finite set or an infinite set(including interval). The definitions are given as follows:

Definition 3 (*Product variety*)**.** Product variety describes a design domain for similar products, these similar products are defined by a number of common design variables and personalized design variables. The common design variables describe the common characteristics of these similar products while the personalized design variables define their individualized characteristics. The value of the design variables could be a finite set or infinite set.

(a) Business stapler (b) High capacity stapler

(c) Smooth grip stapler (d) Mini stapler

Figure 1-2: Four stapler variants

Definition 4 (*Product family*)**.** A product family represents a technical domain for similar assemblies, which is further decomposed into architecture and attribute sets. Architecture limits the range of variety components used to construct the functional and physical structure of a product variant, while the attribute sets restrict the attribute values of the variety components. An product variant is derived by choosing a set of variety components from the architecture and the values for the attributes of these components from the corresponding attribute sets.

According to definition 3 and definition 4, the similar products have not only similar design specifications, but also similar manufacturing processes. Each individual within a product family/product variety is called a product variant. For example, the four staplers shown in figure 1-2 can be grouped into a product family based on definition 4. These staplers share a common component structure, like handle, carrier, pusher, anvil and base, and all of them serve a main common function that is inserting the staples into the papers. However, the shapes and dimensions for the components of these staplers could be different; for instance, the handles of all these staplers have different shape and dimension. Some stapler variants can share common components, for example, the carrier used in stapler (a) and that in stapler (c) could be exactly the same. The assembly processes to fabricate these stapler variants are also similar. By configuring the variety components and the specifications of the attribute values for the variety components , more staple variants could be created.

Similar to product variety and product family, the concepts of part variety and part

family are also extended with the concept of "domain" to describe infinite set of similar parts from functional point of view and from manufacturing point of view respectively.

Definition 5 (*Part variety*)**.** Part variety describes a design domain for similar parts, these similar parts are defined by a number of common design variables and personalized design variables. The common design variables describe the common characteristics of these similar parts while the personalized design variables define their individualized characteristics. The value of the design variables could be a finite set or infinite set.

Definition 6 (*Part family*)**.** Part family represents a part domain which is further decomposed into architecture and attribute sets. A part variant can be derived by choosing a set of variety components from the architecture and the values for the attributes of these components from the corresponding attribute sets.

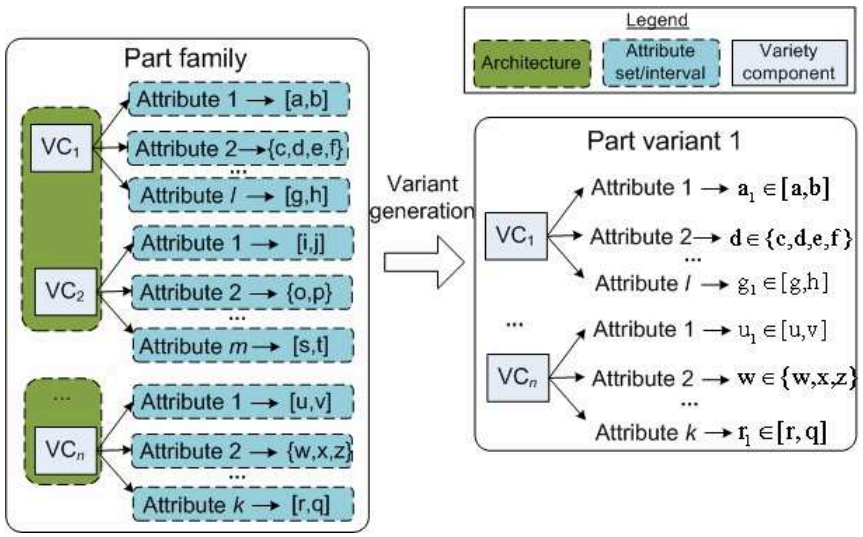


Figure 1-3: Illustration of the concept of part family

An illustration of part family is shown in figure 1-3. Architecture provides the variety components that a part family can provide to its variants. Variety components could be the functional components and the physical components which are used to construct a part variant. For instance, the technical functions of the part variants are a set of variety components. Some variety components can have a set of attributes. Attribute domains limit the values that can be chosen for an attribute of a variety component, such as the diameter of a hole feature, the depth of a pocket feature. The domains in a part family could be either a finite set or an infinite set. With this definition, a part family could have infinite

number of part variant solutions.

The parts of a product are designed to answer the question that how they can work together to deliver the final functions of the product. From the perspective of Function-Behavior-Structure, the structure of a part is determined by its behaviors required in a product, and its behaviors depend on the technical functions it serves in the product (Qian and Gero, 1996). A part family is defined corresponding to a product family. The similar parts from different product variants of a product family are designed as a family. Each individual within a part family is called a part variant. For example, the handle bodies of the four staplers in figure 1-2 can be designed as a handle bodies family, since they fulfill the same technical functions: 1) hosting the hammer part and the carrier part; 2) connecting with hammer, carrier, base and pin, and all of these handle bodies have a pocket feature to accommodate their hammer and a hole feature to work with their pin.

Manufacturing processes for a product/part determine how to transform the raw material into a final product/part. These manufacturing processes can be generally classified into eight categories, as shown in figure 1-4. To limit the research scope, only assembly processes and machining processes are considered, which means only mechanical products that can be produced from assembly processes and parts that can be produced from machining processes are the research objects of this thesis.

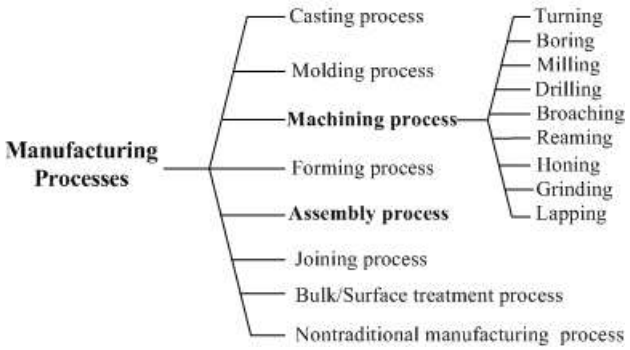


Figure 1-4: Classification of various manufacturing processes (Kalpakjian et al., 2009)

Definition 7 (*Manufacturing process planning*)**.** Manufacturing process planning consists of a sequence of planning activities that define in details the process to transform raw material into the desired form before putting raw material into production. The involved planning activities include: interpretation of design specifications, selection of manufacturing processes, selection of manufacturing tools, determination of process parameters, determination of op-

eration sequences, and cost calculation. The output of manufacturing process planning is manufacturing process plan.

The specific activities and methods involved in manufacturing process planning vary according to the type of manufacturing process. As only assembly process and machining process are considered, this research only dedicates to address the issues of RPP related to Assembly Process Planning (APP) and Machining Process Planning (MPP).

According to the level of detail, manufacturing process planning has two granularities, conceptual process planning and detailed process planning, as shown in figure 1-5. Conceptual process planning focuses on the planning activities including: interpretation of design specifications, selection of manufacturing processes, determination of operation sequences and cost calculation, while detailed process planning provides detail process information to embody the macro process plan including the information about manufacturing tools, process parameters, and even NC code (ElMaraghy, 1993; Srinivasan and Sheng, 1999; Chaube et al., 2012). The output of conceptual process planning is the input of detailed process planning and the output of detailed process planning-detailed process plan is the final process plan which can be executed by a production system. One advantage to divide process planning into two granularities is conceptual process planning can be done without giving available manufacturing resource information, and then the conceptual process plans can be detailed or modified by detailed process planning once the resource information from product system is available, which is beneficial for the implement of concurrent engineering.

Definition 8 (*Production planning and scheduling*)**.** Production planning and scheduling is a set of planning activities that determines the allocation of available production resources over times to a set of tasks defined in manufacturing process plans so as to best satisfy some criteria.

Production scheduling can be described as a mathematical scheduling problem: Given a set of n tasks which are to be processed on m machines with defined technological constraints for each job, find a sequence in which jobs pass between machines such that the technological constraints are satisfied and the resource allocation is optimal with respect to some performance criteria (Sormaz, 1994).

The relationships between product design, process planning and production scheduling in a computer integrated manufacturing system are shown in figure 1-6. Process planning

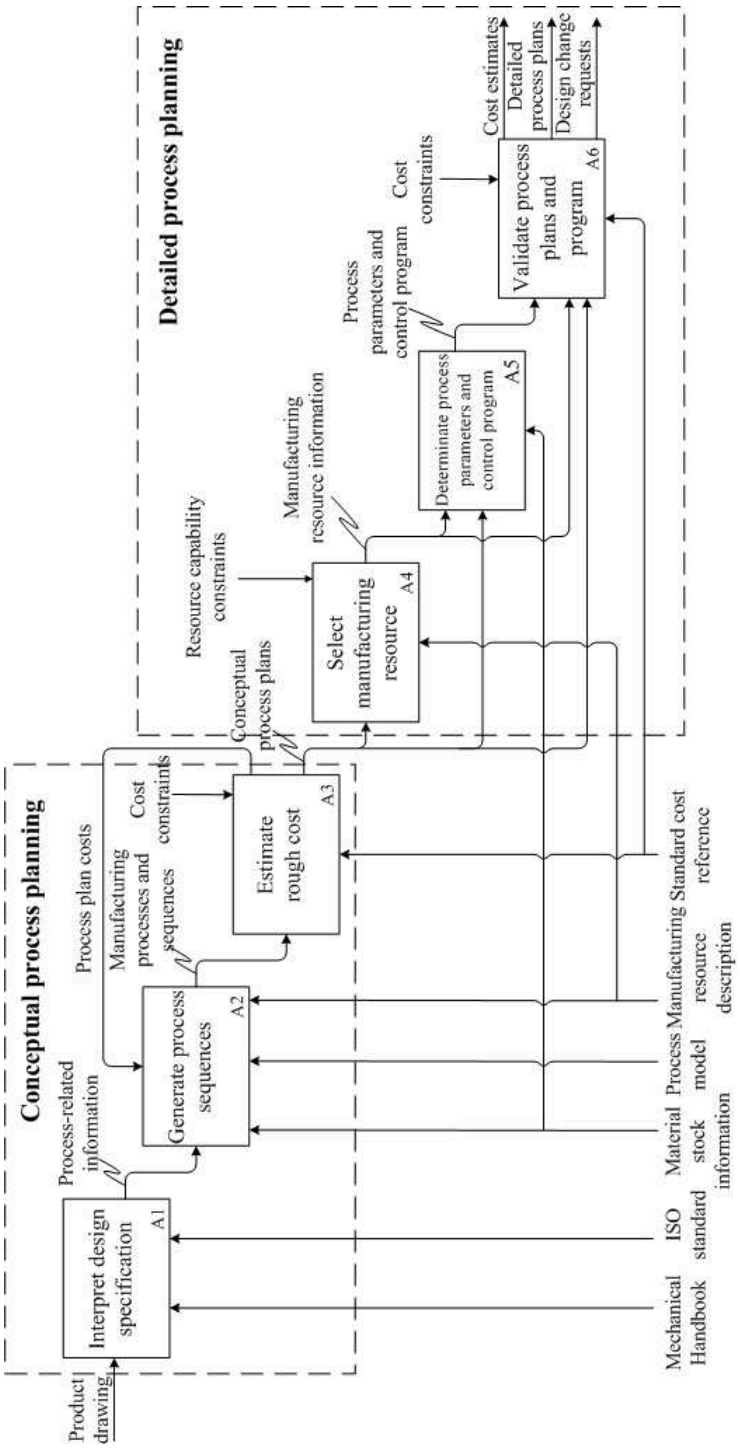


Figure 1-5: Activity diagram for manufacturing process planning (Scallan, 2003)

is an essential link between product design and production scheduling. It transforms product design information into executable process plan for production. In the manufacturing paradigms of MC and co-evolution of P^3S, product design shifts from individual product

design to product variety design, and production system turns away from "low-variety-high-volume" and "made-to-inventory" mode to "high-variety-low-volume" and "made-to-order" mode. These transitions require process planning not only to be able to flexibly generate the manufacturing process plans for different product/part variants, but also to be capable to agilely respond to the dynamics in the production system.

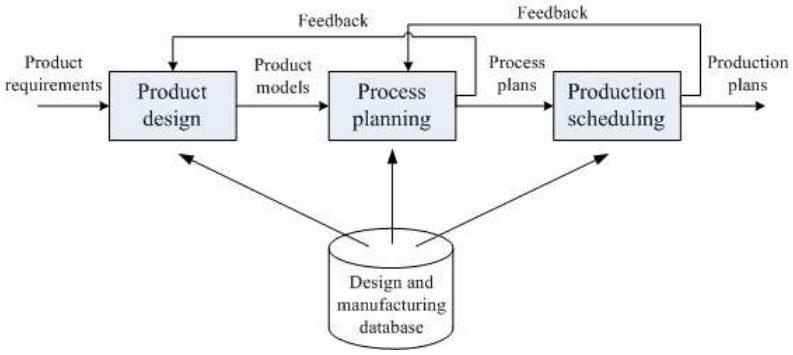


Figure 1-6: Relationships between product design, process planning and production scheduling(Feng, 2003)

Reconfigurable process planning is a promising process planning method for product variety. In accordance with the concepts defined in this subsection, this research investigates the RPP methods on APP for product family and MPP for part family with emphasis on their capabilities to co-evolve with product design system and production system. In the following subsection, the work in the literature related to our subject is reviewed and concluded.

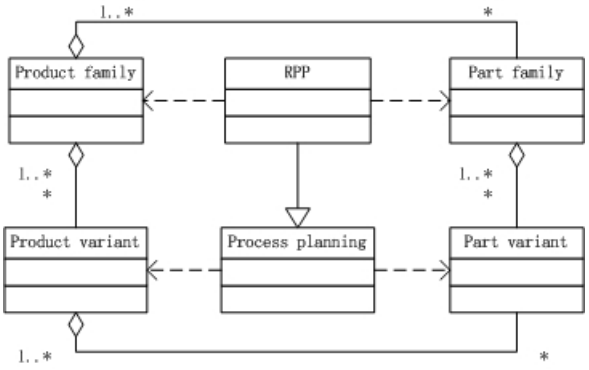


Figure 1-7: Proposed UML class diagram for the relationships between the key concepts

The relationships between the key concepts of this thesis are illustrated through an Unified Modeling Language (UML) class diagram shown in figure 1-7. Part variant has an aggregation relation with product variant, because a product is composed of a set of parts.

The same relation also exists between product family and part family. The aggregation relations also exist between product family and product variant as well as between part family and part variant. RPP has dependency relations with product family and part family, because RPP needs the information from product family and part family for the generation of reconfigurable process plans. The same relations also found between process planning and product variant or part variant. As RPP is a new process planning method for product variety, RPP has a generalization relation with process planning.

1.2 Related work

1.2.1 Product modeling for manufacturing

The main objective of product modeling is to provide computer-interpretable product design information to support various activities in product life-cycle including design, manufacturing, sale, and purchase. Product modeling concerns the issues on what information about product is required and how the information can be represented in a computer-interpretable way. The importance of product modeling has been widely recognized by the research and application communities over the past decades. Abundant product models have been developed for different purposes in computer integrated manufacturing system (Krause et al., 1993; Fenves, 2001; Gujarathi and Ma, 2011; Chen et al., 2012). Here, we mainly focus on product modeling for assembling and for machining.

Bill of materials

Bill-of-Materials (BOM) describes the component structure and number of components of a product, which is one of the earliest tools to represent product data for production control.A BOM generally consists of three parts of elements as described below(Garwood, 1988):

- *Items.* They are the components of a product. An item could be a part or a sub-assembly. These parts and sub-assemblies of a product could be outsourced from other companies, or could be manufactured by the company who produces that product.

- *Goes-into relationships.* They are the linkages between a parent item and a child item. In a BOM, a parent item is usually connected to a set of child items by several goes-into relationships. A goes-into relationship linking a parent item to a set of child items

which means that this parent item is assembled from these child items.

- *Attributes.* They are the data describing the characteristics of items and goes-into relationships. The attributes related to items could be item code, item description, type of item, cost price, lead time; the attributes related to goes-into relationships could be parent item number, child item number, sequence number, quantity/per, and scrap factor.

Figure 1-8 shows a BOM of a desktop stapler. The items presenting in this BOM consist of the parts and sub-assemblies of this product. These items are shown with code numbers for identification. The goes-into relationships connect the parent items at upper level with the child items at lower level. Along with the goes-into relationships, there are two attributes, one describes the assembly sequence number, and another one represents the quantity number of child items per parent item.

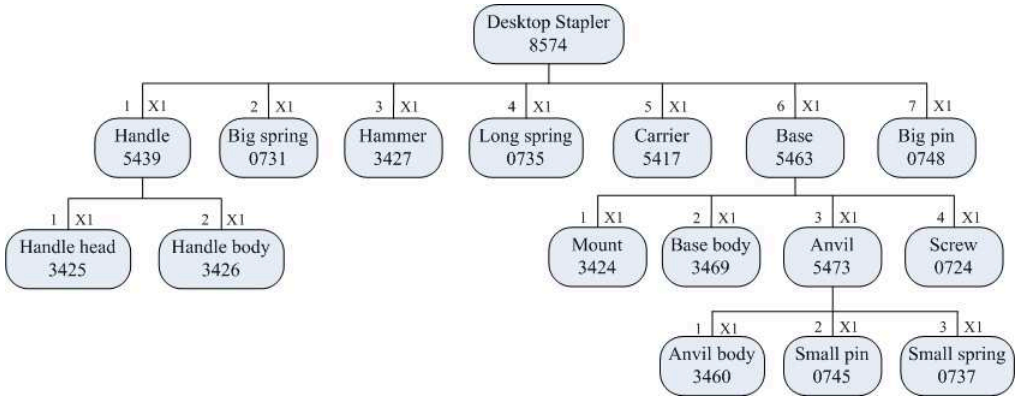


Figure 1-8: Bill-of-material structure of a desktop stapler

Generic bill-of-material

As the number of product variants increase in the company, traditional BOM become inflexible and cumbersome to manage large amounts of product variants data. In order to reduce redundant structures in traditional BOM, Hegge and Wortmann (1991) introduce a new BOM model, named Generic Bill-of-Materials (GBOM), which can specify all variants of a product family in one BOM. In a GBOM, all part variants of a part are represented by a primary generic part; all variants of a subassembly are represented by a generic subassembly product; attributes are defined for each primary generic part and generic subassembly product, and different attribute values can be allocated to these attributes to specify product

and component variants.

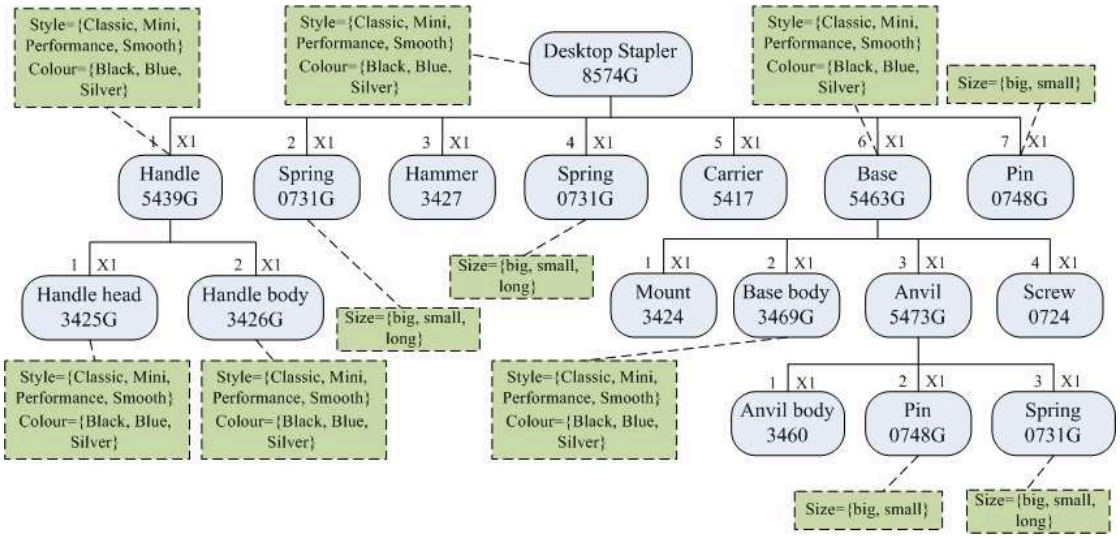


Figure 1-9: Generic bill-of-material of a desktop stapler family

Figure 1-9 shows a generic bill-of-material of a desktop stapler family. Some of the items in this GBOM are generic items which represent a set of variants, instead of a single variant in the traditional BOM. Along with these generic items, the attributes are defined and different values can be taken for these attributes to specify a particular variant. An obvious shortcoming of GBOM is that the product data represented in GBOM are only suitable for material planning purposes, but not for assembly.

Bill-of-Materials-and-Operations

In order to integrate BOM with production control, Jiao et al. (2000) develop a product data model, referred to Bill-of-Materials-and-Operations by combining the BOM structure with the bill-of-operation into a single one. In a Bill-of-Materials-and-Operations (BOMO), a component material is associated with the relevant operation in a bill-of-operation for producing its parent component, and a part kitting process is added to each operation to establish the material requirement link between BOM data and bill-of-operation data. By drawing on the experience of GBOM, BOMO can be extended to generic BOMO to deal with variants resulting from both product changes and process variants by predefined parameters for each generic items, generic goes-into relationships and generic operations. Explosion and planning rules, which are defined in terms of constraints among parameter values, can be used for characterizing both product and operation variants.

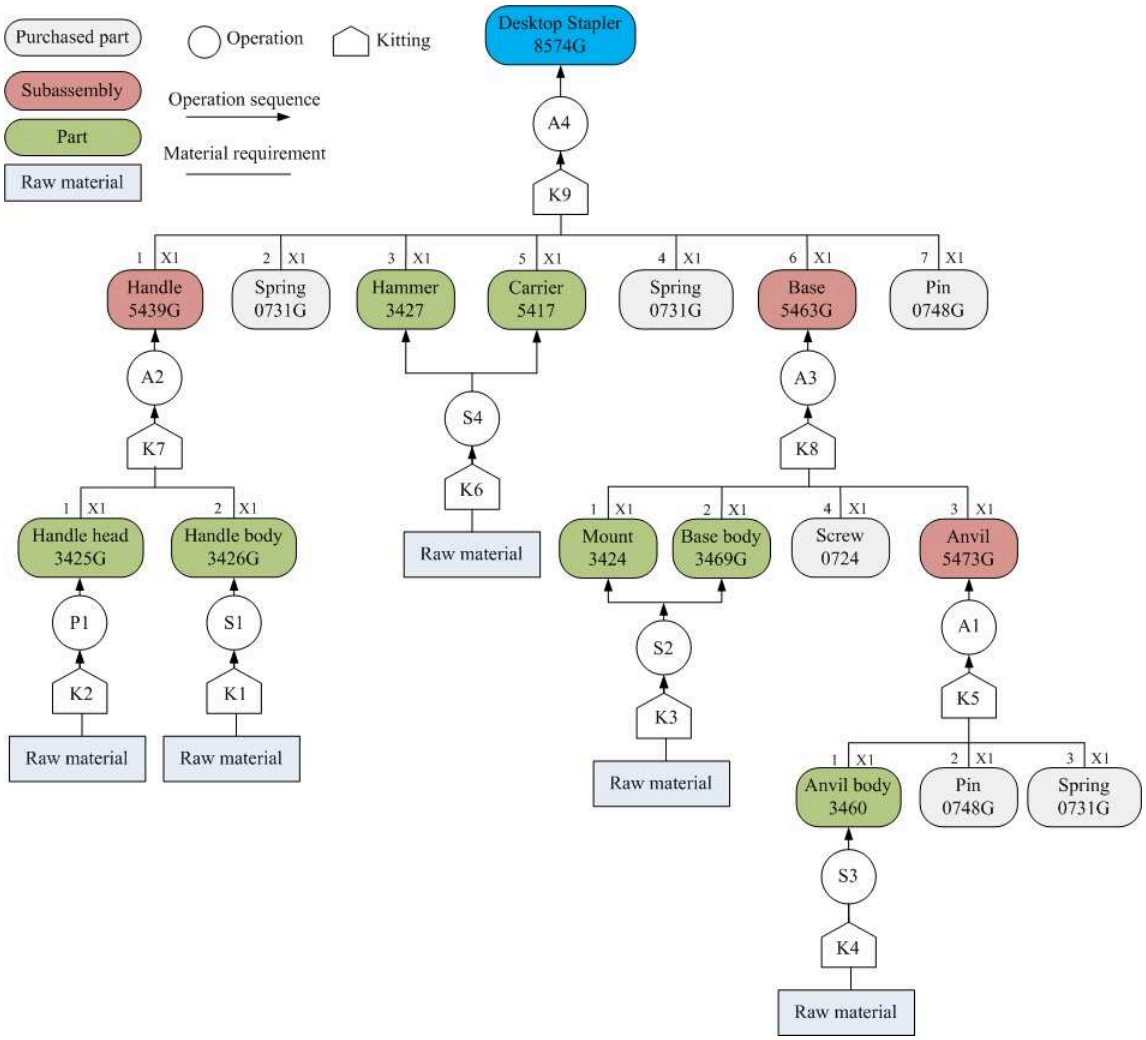


Figure 1-10: Generic bill-of-material-and-operation of a desktop stapler family

Figure 1-10 shows a generic bill-of-material-and-operation structure of a desktop stapler family. As illustrated in the figure, the GBOM of this product family is combined with the bill-of-operation. Operations are connected to the corresponding self-made items including intermediate parts, subassemblies, and the final product. According to the characteristics of self-made items, different types of manufacturing operations are chosen, assembly operations are chosen for subassemblies and the final product, sheet metal forming operations are selected for sheet metal parts, and mould forming operations are used for making plastic parts. A kitting process is attached to each operation, which represents the necessary material preparing process for the operation.

Although generic BOMO can be used to represent product-material relationship and operation-material relationship for a set of product variants, one disadvantage of this mod-

eling method is that it has to assume all the product variants of a product family have the same product structure and manufacturing process structure, which is not always true in a dynamic and evolved product system.

Liaison graph

Being different with BOM, a liaison graph is used for representing connectivity among the parts of a product (Henrioud and Bourjault, 1992). This modeling method uses nodes to represent parts and uses lines between nodes to represent liaisons or connections between parts. Figure 1-11 shows a liaison graph of a desktop stapler.

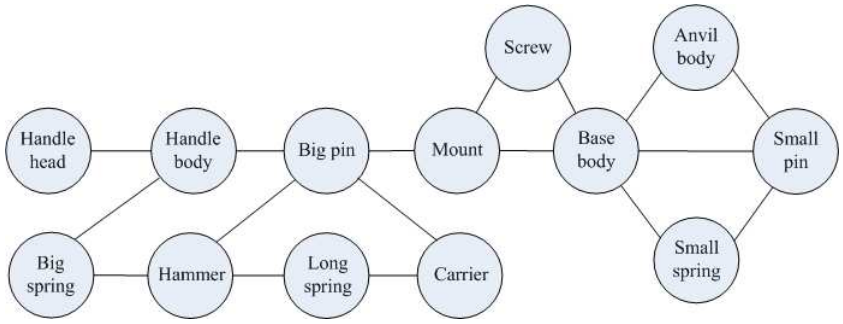


Figure 1-11: Liaison graph of a desktop stapler

Liaison graph plays an important role in assembly sequence analysis. It specifies necessary part connecting constraints for APP. Along with feasibility judgment of assembly, liaison graph can be used to determine the feasible assembly sequence for a product. One method proposed by de Mello and Sanderson (1991a) applied graph theory methods on processing liaison graph and enumerate all the cut sets that can be made through it. The cut sets are then tested for feasibility of assembly. A feasible assembly process can be derived after repeatedly decompose feasible cut sets. de Mello and Sanderson (1991a) also extended liaison graph into a relational model which is a 5-tuple $< P, C, A, R, a - function >$ where:

- P is a set of parts in the product;
- C is a set of contacts between surfaces of two parts in the product;
- A is a set of attachment acting on a set of contacts;
- R is a set of relationships between the elements in P, C and A;

- $a - function$ s a set of attribute functions whose domains are subsets of subsets of $P \cup C \cup A \cup R$;

The relational model can be represented by a graph and a text on the associated attribute functions. Figure 1-12 shows a relational model graph of a desktop stapler. Given a relational model of a product, the liaison graph of this product can then be generated by defining a rule on the relational model, the liaison graph, <V, E>, of the product satisfies the following conditions:

1. $V = P$;
2. $E = \{(p_i, p_j) \mid (p_i \in P) \wedge (p_j \in P) \wedge \exists c(c \in C \wedge \{p_i, p_j\} = part(c))\}$

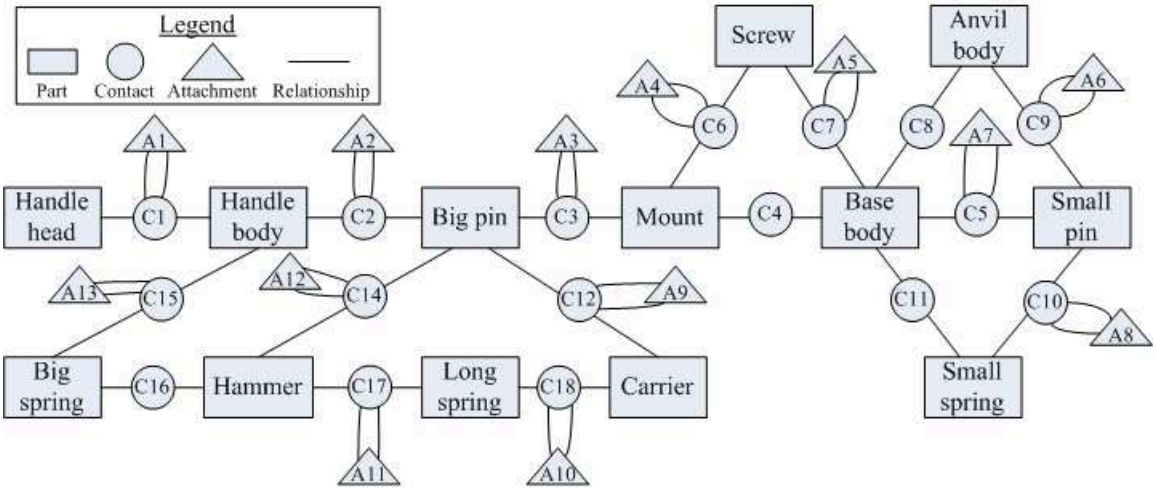


Figure 1-12: Relational model graph of a desktop stapler

A drawback of this relational model is that the information involved in this model could be very large, especially when considering a complex product consisting of hundreds of parts or a product family.

De Lit and Delchambre (2003) took product family into account and proposed a concept of generic liaison. They firstly gave a definition to a concept of generic component representing a set of primitive elements fulfilling the same functions in different product variants of a product family, and then a generic liaison was defined as a link between two generic components whose component variants have at least one liaison relationship.

Figure 1-13 illustrates a generic liaison between two generic components of a stapler family. One generic component is a generic handle head that contains two handle head

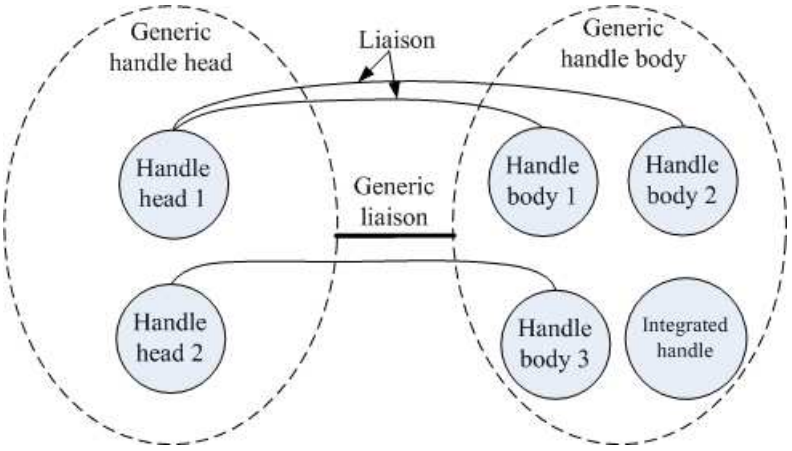


Figure 1-13: Generic liaison between two generic components of a stapler family

variants, another generic component is a generic handle body that consists of four handle body variants corresponding to the two handle head variants. Based on the definition, there exists a generic liaison between generic handle head and generic handle body. Generic liaison method provides a shortcut to represent the linkage relationships between the components of product variants in a product family so as to enable processing product data at product family level, instead of dealing with the data of every product variant in the family.

However, this method does not provide a mechanism to track back the liaison relationship between two components of a particular product variant in the product family from the generic liaisons.

Feature-based models

Feature-based model has long been recognized as an object oriented technology to encapsulate the manufacturing significance of the geometrical specifications of a part or product(Shah, 1991a,b; Allada and Anand, 1995). The definition of feature could have different explanations depending on its application scenario. From a manufacturing point of view, feature is defined as manufacturing feature which is a collection of related geometric elements used to describe an indecomposable object that correspond to a particular manufacturing method or process. Manufacturing features for reasoning about the assembly processes are assembly features; while that for representing machining-related design specifications are machining features.

In light of feature based model, manufacturing process planning knowledge can be defined to map manufacturing features to manufacturing processes without involving pure geometric

model. Over the past decades, large amounts of research efforts have been devoted into developing models and frameworks both for assembly features and for machining features.

By considering the incompleteness of liaison graph in representing the mating relationships of the parts of an assembly, Case and Wan Harun (2000) introduced feature relation graph and face mating graph to provide detail assembly information between the parts of an assembly. Their research was limited to establish a representation but the method to analyze the proposed model for generating the assembly processes was not given.

Shah and Rogers (1993) investigated the assembly relationships at three levels: sub-assembly level, part level, and feature level. They found that the same basic relationship structure can be applied to the three modeling levels, which proves the feasibility to extend the feature paradigm to assemblies. Four abstract levels were defined in the proposed assembly structure, including sub-assembly, part, form features and simple volumes. The constraints for the structure relations of form features and the assembly relations were specified through a series of predefined generic control elements.

Deneux (1999) introduced an assembly-feature-based approach for supporting intelligent design and manufacturing of complex assemblies. Although some guidelines were provided by Deneux to show how assembly features can be characterized in any domain of application, a detail model structure of assembly feature was not given.

Van Holland and Bronsvoort (2000) proposed an object-oriented feature-based product model using assembly features to represent specific assembly information. They showed that feature is a valuable concept which can be used to integrate not only single-part and assembly modeling, but also assembly modeling and APP. However, the proposed assembly model was conceptually designed, a complete and detail case study was not found in their research.

Eng et al. (1999) found the weakness of using geometrical reasoning technique alone in assembly process plan. So they used a set of mating features to describe a mechanical product, and defined feature matrices to establish the degrees of freedom between the mating features. Based on the feature matrices, a kinematic pair liaison diagram for an assembly can be defined to fulfill a series of testing procedures for finding the feasible assembly sequence. Some researchers put their focuses on the aspect of the implementation of a feature based modeling system.

Mascle (2002) developed a product assembly features system using multi-agents system

structure. Li et al. (2010) developed a rapid assembly modeling system using predefined typical assembly features. Zhang et al. (2015) proposed a concept of assembly feature pair to represent the assembly behavior of two mating features in part models. By embedding the model into part models at part modeling stage, a part can be pre-assembled with its potential mating parts. The model was implemented in a prototype system for assembly modeling and simulation integration.

Some research efforts have also been put on the standardization of feature-based assembly representation. Such efforts include ISO 10303 for computer-interpretable representation and exchange of product data (SCRA, 2006) and the Core Product model and the Open Assembly Model proposed by US National Institute of Standards and Technology (Fenves, 2001).

Assembly feature model has been proved to be an effective way to represent the assembling information between two mating parts in a product. It can provide the necessary information for the activities in APP, including degree of freedom analysis, assembly feasibility analysis, assembly stability analysis, assembly collision analysis, assembly operation selection, and assembly sequencing. Although large amounts of researches have been done in feature based assembly modeling, until now there is no universal assembly model for APP. In addition, most of the existing researches study assembly feature model for a single part, there is few efforts on assembly feature model for a set of similar part variants in a product family.

Different with assembly feature, machining features are applied to represent the design specifications of features in a part in order to provide the necessary information to MPP. The necessary information includes geometric shape and dimension, surface quality, tolerance, tool approach direction and interaction relationships with other features.

In order to representing the necessary information of features, many researchers and organizations have developed their own feature models for different purposes. Catania (1991) proposed a form-feature-based model for a CAD/CAM environment, but the model was limited to the geometrical representation of machining features. Gu (1994) defined a surface-based design representation scheme, the scheme uses a limited number of features to provide a general model space for a variety of parts and products, but only an inspection planning application was demonstrated, no application for MPP was given. Case (1994) introduced a feature-taxonomy-based feature model, which uses External Access Direction to represent

potential directions for machining. In order to capture the information on Geometric Dimension and Tolerance (GD&T) of a part, Shah et al. (1998) developed a GD&T model which is based on relative degrees of freedom of geometric entities. Based on this model, the GD&T of machining features can be automatically determined by feature recognition. Gonzalez and Rosado (2004) proposed a detailed feature-based product information model to represent all of the part information in a manner tailored to the needs of process planning without using any geometric entity. Amaitik and Kilic (2005) developed a STEP-based feature modeler for prismatic parts. Li et al. (2006b) proposed a feature-based neutral representation of part geometric information in a web-based parts library. A comparison between these different models is given by table 1-2.

Table 1.2: Comparison among different part feature model in the liturature

Criteria	Catania (1991)	Gu (1994)	Gonzalez and Rosado (2004)	Amaitik and Kilic (2005)	Li et al. (2006b)
Involving geometric entity	√	√	-	√	√
Using feature taxonomy	√	√	√	√	-
Supporting machining feature	√	-	√	√	-
Tolerance Information	√	√	√	√	√
Machining accessibility	-	-	√	√	-
Feature Interaction	-	√	-	-	-
SETP compatible	-	-	√	√	-

It can be drawn from the existing researches that feature-based modeling is an effective approach to represent the necessary information of a part both for APP and for MPP. However, one difficulty for the application of the existing feature-based modeling approaches is that there is still no universal feature-based model which can be used for both APP and MPP, so researchers have to develop their own feature model according to their own application scenario. In addition, most of the existing feature-based models are designed to represent the design specifications of one single part; few of them consider the concept of part family.

1.2.2 Configuration technology for product variety and process variety

Configuration is considered as a type of design activity that a technical system being designed is assembled from a set of pre-defined components by complying with certain configuration constraints. Configuration has been proven to be an effective technology to reduce the development and maintenance costs for the implementation of mass customization (Felfernig et al., 2014).

Product configuration is one successful application of configuration techniques. Product configuration enables the rapid generation of new product variants from a set of predefined product components instead of creating them from scratch (Salvador and Forza, 2004a; Zhang, 2014). Many Artificial Intelligent (AI) techniques have been applied in product configuration to automate this complex activity (Yang et al., 2008; Hong et al., 2008). Xie et al. (2005) proposed an approach to model and solve an engineering product configuration problem based on the constraint satisfaction paradigm. Yang et al. (2008) adopted an ontology language and a rule language to model product configuration knowledge. Some researchers also use genetic algorithms (Li et al., 2006a; Hong et al., 2008) and case-based reasoning (Tseng et al., 2005) to solve product configuration problem.

One immediate effect of product variety is the growth in number of process variants. With a witness of the effectiveness of product configuration techniques, some researchers have attempted to apply configuration technique on process variant generation.

Schierholt (2001) was one of the pioneers who proposed a concept of process configuration in analogy with the concept of product configuration. The proposed process configuration approach uses a plan skeleton for describing the knowledge of the process plan family and the process configuration problems are solved as constraint satisfaction problems.

Aldanondo and Vareilles (2008) also explored the possibility to extend product configuration towards process configuration. They defined process configuration problem in a way of constraint satisfaction problem:

- Hypothesis: a routing is a set of operations linked with anteriority constraints, an operation is a set of resources characterized by a required quantity;
- Given: (1) a generic model of a configurable routing able to represent a family of production processes with all possible variants and options, (2) a set of inputs, where an input corresponds with a selection of an operation, a resource or a quantity value;

- Solve: a routing configuration problem that finding at least one set of operations with relevant sets of pairs that satisfies all the constraints and the inputs.

To design a decision support mechanism that configures process variants corresponding to a given product configuration, Jiao et al. (2004) developed a process variety model using object-oriented Petri nets with changeable structures.

Zheng et al. (2008) proposed a systematic knowledge model in order to facilitate process configuration; their model is categorized into 6 levels: (1) core process skeletons, (2) process networks, (3) process routes, (4) process segments, (5) processes/work plan, and (6) operations/working step; then a rule-based approach was put forward to achieve rapid process configuration by reusing the process knowledge.

Pitiot et al. (2014) studied the use of an evolutionary optimization algorithm called constraint filtering based evolutionary algorithm to solve an interactive product configuration and process planning problem.

Wang et al. (2015) mapped process configuration problem to a generative constraint satisfaction problem.

To summarize, large amounts of researches have been devoted on the configuration technology for product configuration, while the configuration technology for the generation of process variants is drawn less attentions. Moreover, most of the existing researches on process configuration choose product family as their focuses and no reference can be found on machining process configuration for a part family.

1.2.3 Manufacturing process planning

Manufacturing process planning defines all necessary steps and the parameters in these steps to instruct the manufacturing system to generate the required shape, properties, quality of a part or a product, within the given constraints while optimizing some criteria. With the widespread computer application system, traditional manual process planning approach has been assisted by Computer-Aided Process Planning (CAPP) and Computer-Aided Assembly process Planning (CAAP). CAPP and CAAP are two research domains with abundant outcomes including models, algorithms and systems. Since this thesis focuses on manufacturing process planning for product/part variety, the relative work on process planning in the literature is synthesized with the emphasise on the models and approaches for process planning for product/part variety.

Computer-aided process planning

MPP consists of a sequence of planning activities that define in detail the process elements to instruct manufacturing system to remove material from raw material so as to obtain the desired form of a final part. The process elements determined by MPP include machining processes, operations, resources and relevant parameters. According to the level of detail, it has two granularities: 1) conceptual MPP which focuses on process selection and feature sequencing considering design constraints; 2) detailed MPP which concerns the optimal process parameters to satisfy manufacturing constraints (Xu et al., 2011).

With the widespread application of information technology, traditional manual MPP approach has been assisted by CAPP. The existing CAPP approaches can be generally classified into three categories: 1) variant approaches; 2) generative approaches, and 3) hybrid approaches. Variant approaches were adopted by the early stage of MPP systems (Zhang et al., 1984), while recent researches concentrate on the generative approaches and hybrid approaches (Nonaka et al., 2013).

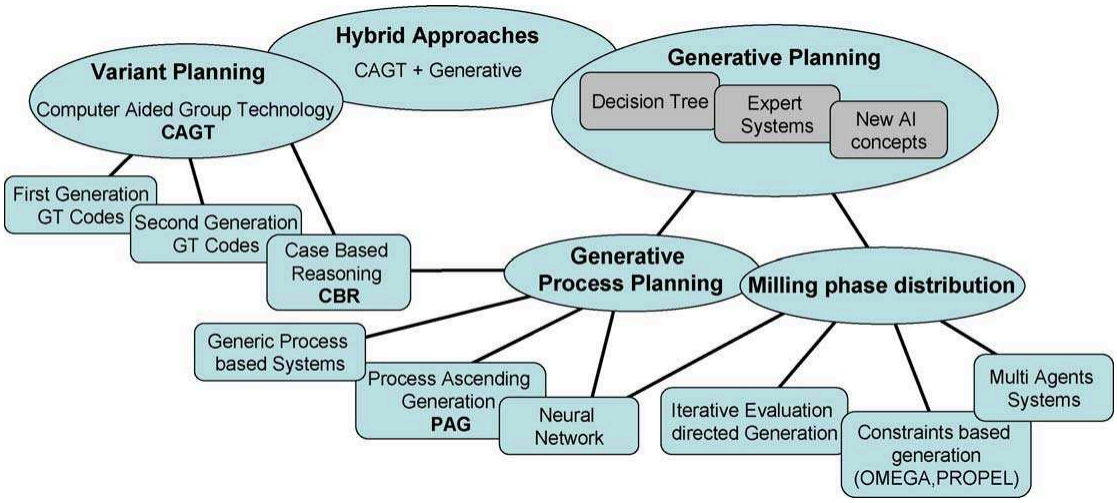


Figure 1-14: Techniques used in the three CAPP approach(Etienne et al., 2006)

Many technologies from the field of computer science have been used to automate the activities in MPP, including: 1) Feature-based technologies (Sormaz and Khoshnevis, 2000; Wang et al., 2006; Givehchi and Wang, 2015); 2) Knowledge-based technologies (Liu and Wang, 2007; Denkena et al., 2007); 3) Artificial-intelligent-based technologies (Barrabes and Villeneuve, 1993; Qiao et al., 2000; Liu et al., 2013); 4) Standard-compliant technologies (Xu et al., 2006; Chung and Suh, 2008); and 5) Internet-based technologies (Agrawal et al.,

2009; Wang, 2013; Hu et al., 2008).

A few researchers consider the impact of part variety in their approaches for MPP. Mäntylä and Sohlenius (1993) developed a feature-based part family model which is used to contain process plan specifications. Once a new part has been generated as an instance of the part family, information on the instance can be propagated to the corresponding process plan specification so as to yield an actual process plan. The model is implemented in a Manufacturing Cell Operator's Expert System which combines variant process planning and generative process planning in a single framework.

An old approach to generate machining process plan for new part variant is the reuse of the existing plans, such as group-technology-based approach (Burbidge, 1993) and case-based reasoning approach (Markus et al., 1997). However, this kind of approaches is lack of flexibility and adaptability in terms of new design changes on part variants. In order to overcome the shortcoming of the old approach, ElMaraghy (2006) introduced a new concept of process planning approach - Reconfigurable Process Planning (RPP). RPP is recognized as an important enabler of changeability for evolving products and manufacturing systems (ElMaraghy, 2009). Azab et al. (2007) proposed a hybird MPP approach for RPP. In Azab's method, a composite part is used to represent a part family and the features/operations precedence graphs of this composite part are used to represent precedence constraints in all part variants, then the precedence graphs of a specific part variant is derived by modifying the precedence graphs of the composite part. Azab et al. (2007) further developed this method by giving a mathematical model and formulation, the reconfiguration of precedence graphs is achieved by inserting/removing features iteratively using a 0-1 integer programming model. Although Azab's method can generate the new process plan for a new part variant without generating it from scratch, generation of the precedence graphs for the composite part could incur a large computation burden.

Computer-aided assembly planning

CAAP helps engineers find the feasible and optimal assembly operation sequences for an assembly or a subassembly as well as the necessary assembly resources and fixture plans. One aspect in CAAP related to our research is assembly plan model which defines how the process elements in an assembly plan are represented and organized. Various assembly plan representation models have been put forward in the literature (Jiménez, 2013). Most of

them are graph-based representations in which the nodes are used to denote assembly tasks or assembly states or subassemblies or assembly connections, the edges are used to indicate the precedence.

In general, assembly plan models can be classified into two categories: explicit models and implicit model(de Mello and Sanderson, 1991b). Explicit representations explicitly express the assembly tasks and the precedence among those tasks. The explicit models include order list, binary tree(Wolter, 1991), directed graph (de Mello and Sanderson, 1991b), and AND/OR graph (Thomas et al., 2003). Figure 1-15, 1-16, 1-17 show examples for binary tree, directed graph and AND/OR graph. Implicit models are based on establishment conditions and on precedence relationships that implicitly constrain the feasible assembly sequences. Bonneville et al. (1995) encoded precedence relationships as geometric constraints, which express the absence of collision-free trajectories that allow two subassemblies to contact. Gu et al. (2008) used an ordered binary decision diagram to decrease the storage space of the feasible assembly process sequences.

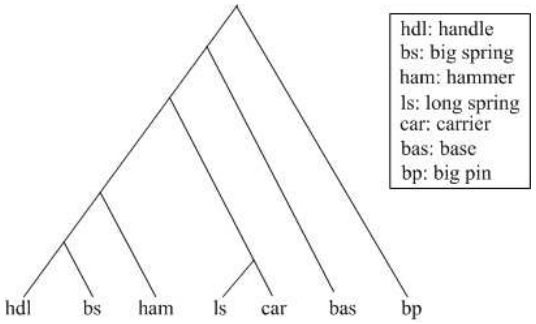


Figure 1-15: Binary tree representation for the assembly sequences for a stapler

Another relevant aspect in CAAP is assembly sequence planning. Many algorithms have been developed for the generation of feasible and optimal assembly sequences based on a specific assembly sequence model. A direct way is to enumerate all the possible subassemblies, and then to test the feasibilities of all the possible sequences between these possible subassemblies. de Mello and Sanderson (1991a)'s cut-set method is an example of this kind of methods. Although the exhaustive method could be sound and complete, as the number of parts in a product increases, this kind method could incur a large computation burden. Some researchers use disassembly-based methods to build assembly sequences reversely (Martinez et al., 2009; Li et al., 2014). In comparison with the straight-forward methods, one advantage of disassembly-based methods is that they will never dead-end.

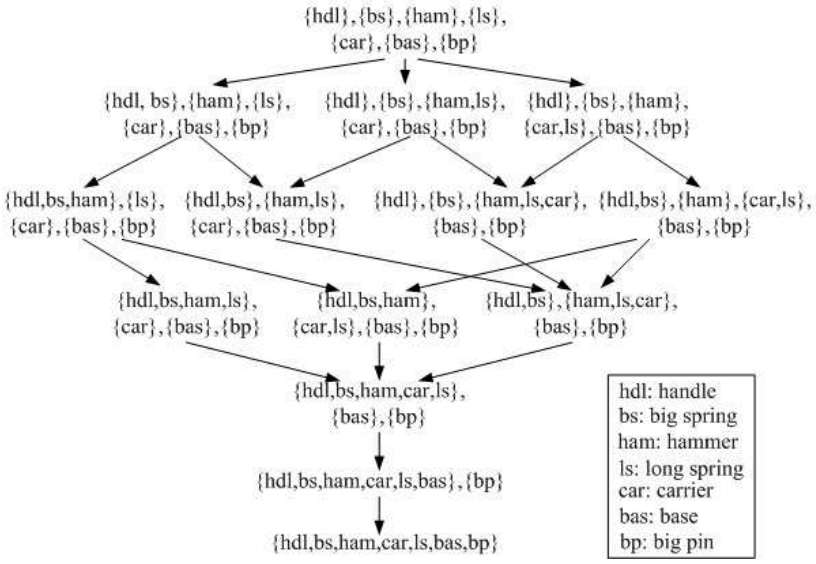


Figure 1-16: Directed graph representation for the assembly sequences of a stapler

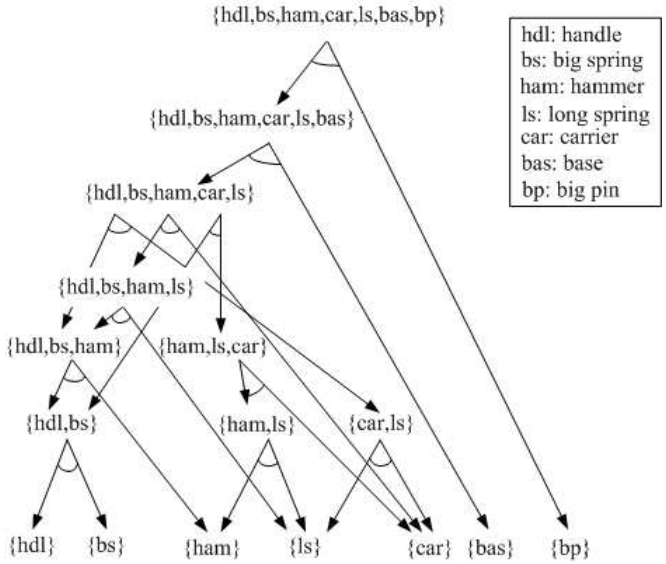


Figure 1-17: AND/OR graph representation for the assembly sequences of a stapler

In order to overcome the shortcoming of the exhaustive methods and improve the efficiency of problem solving, many researchers began to use heuristic algorithms to generate assembly sequences, such as genetic algorithm (Marian et al., 2006), particle swarm algorithm(Wang and Liu, 2010), ant colony algorithm (Wang et al., 2005), and neural networks(Chen et al., 2008). Other CAAP approaches include: knowledge-based methods(Hsu et al., 2011), geometric-based methods(Thomas et al., 2003), graph-based methods(Niu et al., 2003) and constraint-based methods(Zhao and Li, 2009).

In comparison with the research on CAAP for one single assembly, less research efforts are devoted to the approaches for APP for a product family.

Gupta and Krishnan (1998) found the importance of product family-based assembly sequence design for the economic attainment of product variety; their methodology starts with the identification of the maximally generic subassembly in a product family, which is achieved by applying graph-theoretic algorithms on a product family interconnection diagram; and then the assembly sequences for any product variant are generated from components to maximally generic subassemblies, then to the finished product variant.

Martinez et al. (2000) proposed a two-steps approach to generate the assembly process plans for the product variants in a product family: (1) determination and selection of the parent assembly sequences of the product family, then (2) determination of the specific assembly sequence of each product variant from the parent assembly sequences.

Fujimoto et al. (2003) studied assembly process design for product variety and proposed an information-entropy-based approach to strategically manage manufacturing complexities induced by product variety.

De Lit and Delchambre (2003) decomposed a product family into functional entities; and then the assembly plans for the product family were constructed from the precedence graph of the functional entities by analyzing the generic liaison graph of the product family.

The RPP approach proposed by Azab et al. (2007) has been also applied to reconfigurable assembly planning for a household product family (Azab et al., 2009). In the method the operation precedence graph for a new product variant was generated by removing the unnecessary operations from the original operation precedence graph and inserting new operations to it.

Kashkoush and ElMaraghy (2015) proposed a mixed-integer programming model to find the optimal consensus assembly sequence tree for an existing product family based on the assembly sequence trees of individual product family members. The generated consensus tree serves as a master assembly sequence tree from which the assembly sequence of new product variants can be generated.

One common characteristic in the existing research on CAAP for product family is that generic or master models are defined to represent the commonality of product family members. The assembly process plans for a particular product variant are generated by analyzing or modifying these models.

1.3 Discussion

In mass customization, the number of product variants provided by manufacturers increases. As the increase of product variants, the total number of part variants also rises. Therefore, there exist two levels of variety in terms of a mechanical product system: Product variety and part variety. Organizing similar product/part variants into a product/part family can help manufacturers handle the manufacturing complexity induced by product variety. In this thesis, the existing concept of product/part family is extended by using a mathematical definition of "domain" to allow a product/part family to have infinite family members.

Manufacturing process planning is a knowledge intensive and complex task that transforms design information into manufacturing processes and determines the optimal sequence of operations. As a connecting bridge between product design and production, manufacturing process planning plays a key role in maintaining high levels of responsiveness and adaptability while propagating variety from product to process. As various types of manufacturing processes exist in current manufacturing domain and the specific activities and methods involved in manufacturing process planning vary according to the type of manufacturing process, this research limits itself to APP for product family and MPP for part family.

Large amounts of research efforts have been devoted into both MPP and APP. Although they require different process planning knowledge, they have some similarities. The first one is that they both involve three planning steps: 1) Definition of precedence constraints; 2) generation of all the feasible sequences; 3) choice of optimal sequences according to certain criteria; the second one is that they both use feature-based models to represent process-related information. The third one is that they both use graph-based models to represent process sequences. However, current researches on MPP and APP are done separately; few researches consider the relations between them. Moreover, although a few pioneers attempt to investigate manufacturing process planning for product/part variety, most of the existing process planning methods only aim at that for one single product/part.

A product/part model in process planning is a computer-interpretable representation model that provides all the necessary product/part design information to support the generation of manufacturing process plans. Both liaison graph and feature-based model are more effective representation models than BOM-based representation models for process plan-

ning, because they can express the process-related information without involving geometric computation. Currently, many assembly feature models and machining feature models have been given in the literature, but few feature models can be found for capturing the design specifications for both APP and MPP. Moreover, most of the existing feature models focus on representing the information of one single part, the feature model for a set of similar part variants in a product family is seldom considered in the literature.

Product configuration and process configuration are two crucial techniques for the implementation of mass customization paradigm because of their abilities of generating new variants by configuring a set of predefined product or process components instead of creating them from scratch. Comparing with product configuration, less research efforts have been put on process configuration; few articles on process configuration refer to the configuration of process plan.

Table 1.3: Comparison between the related approaches in the literature and the aims of this thesis

Methods	Product/part model	Process plan model	Configuration technology	Process plan generation
Main approaches in the literature	Single, Static	Single, Static	Product configuration	Pre-planning, replanning
RPP method in this thesis	Constantly evolving Family, Dynamic	Constantly evolving Family, Dynamic	Integrated product and process configuration	Configuring

A new concept for manufacturing process planning: "Reconfigurable process planning" is proposed aiming at the process planning for evolving product/part families. As an emerging technology combining manufacturing process planning with configuration technology, it has been identified as a key enabler for the changeable and responsive manufacturing paradigm. However, at present, there is no formal definition, representation model and application framework for reconfigurable process planning, which makes it still far away from industrial application. The need for family-oriented process planning method and the insufficient research on RPP foster the motivation of this thesis. Table 1.3 shows a comparison between the main approaches proposed in the literature and the required characteristics of RPP method. The models for product/part and process plan in RPP method should be dynamic and family-oriented. The configuration technology in RPP can realize integrated product

and process configuration. In addition, the process plans for new product/part variants are generated by configuring the existing process plan resource.

1.4 Conclusions

In this chapter, in order to define the boundary of this thesis, the definitions for the relevant concepts are firstly given. A concept of "domain" is introduced to extend the definitions of product family and part family. With this extension, a product/part family could have infinite product/part variants. After that, the related work in the literature is reviewed from three general aspects: 1) product modeling; 2) configuration technology for product variety and process variety; 3) manufacturing process planning. At last, the evaluation of the existing research and the necessity of this research are discussed.

Chapter 2

Feature-based product variety models for RPP

A computer representation of product family is necessary in order to automatically generate the reconfigurable process plans. The aims of this chapter are to identify the necessary information on product/part families for RPP, and then to propose the representation models for a computational implementation. Feature-based modeling is an effective approach to represent the necessary information of a part both for assembly process planning and for machining process planning. In this chapter, feature-based product variety models are proposed which are capable to represent the configuration-related and process-related design specifications of a product family. The configuration-related information of a product family is represented in Product Variety Decomposition Architecture (PVDA), which combines functional architecture with physical architecture at both product level and part level, and variety configuration constraints between configurable elements in the decomposition architecture are expressed by using a propositional-logic-based representation scheme. The process-related information, including the assembly-related mating information of a product family and machining-related design specifications of the part families inside the product family, is represented by using feature-based models.

2.1 Proposition of Product Variety Decomposition Architecture

2.1.1 Product architecture as a part of PVDA

Product design starts with analysis of market requirements, and their transformations into the functional requirements; finally the physical components are designed to realize these functional requirements. Product architecture expresses functional requirements as a set of modular functional elements, and represents the arrangement of these functional elements and their interconnections via a function structure; it also defines the mapping from functional elements to physical components and the interfaces between interacting physical components (Ulrich, 1995).

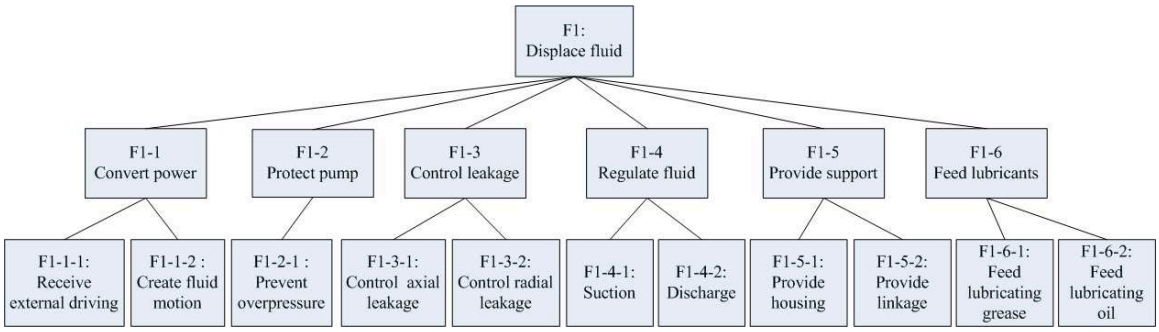


Figure 2-1: Function structure of a gear pump variety

Product architecture is a part of the proposed product variety decomposition architecture. The function structure defined in product architecture forms the first level of PVDA. A function structure is a hierarchical arrangement of modular functional elements deduced from the function requirements. A tree structure can be used to represent the function structure in PVDA as shown in figure 2-1. The function structure of product variety in PVDA is an union from the function structures of all the product variants in the product variety. If $FS_{PF} = \{m_1, m_2, ..., m_i\}$ represents the set of all the functional elements in the function structure of product variety, and $FS_{PV1} = \{n_1, n_2, ..., n_k\}$ represents the set of all the functional elements of a product variant, then:

1. $FS_{PV1} \subset FS_{PF}$;
2. If $m_i = n_k$ and M_i is the set of all the child nodes of m_i, N_i is the set of all the child nodes of n_k, then $M_i \cap N_i = N_i$.

The function structure of product variety is mapped to the physical components of the product variants in PVDA. The mapping happens between the leaf nodes in the function structure of product variety and the physical components of product variants.

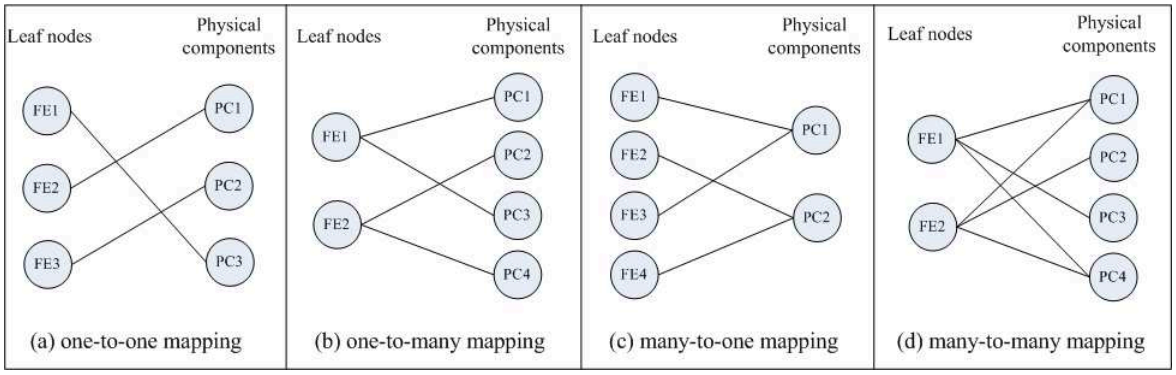


Figure 2-2: Four types of mappings between the leaf nodes and physical components

There exist four types of mappings between the leaf nodes of function structure and physical components:

- One-to-one mapping. It maps only one leaf node to one physical component, as shown in figure 2-2-a. It indicates that the mapped functional element is implemented by only one physical component. Thus, the physical component is a modular component for this functional element;

- One-to-many mapping. It maps one leaf node to multiple physical components, as shown in figure 2-2-b. It means that the mapped functional element is delivered by multiple physical components. In this case, the mapped function can be further divided into multiple sub-functions, each of which corresponds to a subset of the multiple physical components via one-to-one mapping;

- Many-to-one mapping. It maps multiple leaf nodes to one physical component, as shown in figure 2-2-c. It implies that the physical component in the mapping can realize multiple functional elements. In this case, if the physical component is a subassembly, the physical component may be further partitioned into multiple sub-components, each of which relates to a sub-set of the multiple leaf nodes via either one-to-one mapping or many-to-one mapping;

- Many-to-many mapping. The mapping situation which does not belong to the previous three types belongs to many-to-many mapping, as shown in figure 2-2-d. It is the

most complex mapping relation among the four types of mappings. It reveals that one functional element can be implemented by multiple physical components; meanwhile, one physical component can deliver multiple functional elements.

To some extent, whether or not a mapping belongs to one of these four types of mappings depends on the level at which the functional elements and components are considered. For example, if one functional element and one physical component have a one-to-one mapping, and the functional element can be detailed by a set of sub-functions which are the child nodes of the functional element in the functional structure, then obviously the one-to-one mapping become many-to-one mapping between the sub-functions and the component. Similar situations can happen when split a component into a set of smaller sub-components or parts. In addition, particular attention should be paid to one-to-many mapping. Because multiple components work together to deliver one functional element in one-to-many mapping, each component must have their own functions as their reasons of existence. Thus, the functional element in one-to-many mapping can always be divided into sub-functions until there is no one-to-many mapping.If there are only one-to-one mappings in product architecture, the product architecture is considered as a modular architecture; if there are only the rest of the three types of mappings in product architecture, the product architecture is considered as an integral product architecture (Ulrich, 1995). In most cases of mechanical products, the product architecture is a mix of the four types of mappings. A good product architecture should include many-to-many mapping as less as possible.

In PVDA, product architecture represents the function structure for all the product variants. Therefore, the physical components to which the functional elements are mapped come from different product variants. When considering the physical components at parts level, some parts could be shared by the product variants if the product variety is developed on a product platform, while some parts could implement the same key functions and have the similar geometric specifications, but they belong to different product variants; based on definition 6, these parts can be designed as a part family. Moreover, among the parts of product variants, it is possible to have certain parts exclusively deliver the functions for a specific product variant, for example, the parts of a turbocharger are used exclusively to provide extra air into the combustion chamber in a turbo-charged engine while there are no such parts needed in a naturally aspirated engine.

Therefore, in PVDA, the physical components of a product family can be classified into

the following four categories:

- Common parts: they are the parts shared by all the product variants and their design specifications stay the same in all the product variants.

- Part variants in a common part family: these part variants deliver common functions shared by all the product variants, but they could have different design specifications. The part family for these part variants is called a common part family.

- Part variants in an optional part family: these parts variants implement some optional functions only shared by a part of product variants, the part family for these part variants is called an optional part family;

- Personalized parts, they deliver some exclusive functions for a specific product variant or a set of specific product variants, and their design specifications stay the same in the specific product variant(s).

The physical structure of a product family in PVDA represents the arrangement of the four types of product variety components. Part variants are represented by their part families in PVDA.

Figure 2-3 uses a gear pump family as an example to illustrate the mapping relations between the function elements and the physical structure of the gear pump family. It also shows the hierarchical relations between the subassemblies and the parts of the gear pump family. The components of this gear pump family can be found in appendix A and the complete list of its components can be found in appendix B. Figure 2-4 gives the configurable attributes of this gear pump variety. According to the possible combinations of the values of these configurable attributes, 96 product variants can be identified in this gear pump family. In figure 2-3, only 4 of them are illustrated and only partial parts of these gear pump variants are shown. In this example, the common parts of the pump variants include *grease fitting-1* which is used to feed lubrications to the bearing inside the bracket; the common part families are *rotor*, *idler gear*, *gland*, *nut*, *cap screw*, *case*, *bearing*, *bracket*, *gasket*, *screw*. Every pump variant has the part variants from these common part families; the optional part families are *valve body*, *valve spring*, *packing*, *collar*, and *washer*, only certain product variants need the part variants from these optional part families; *grease fitting-2* for feeding lubrications

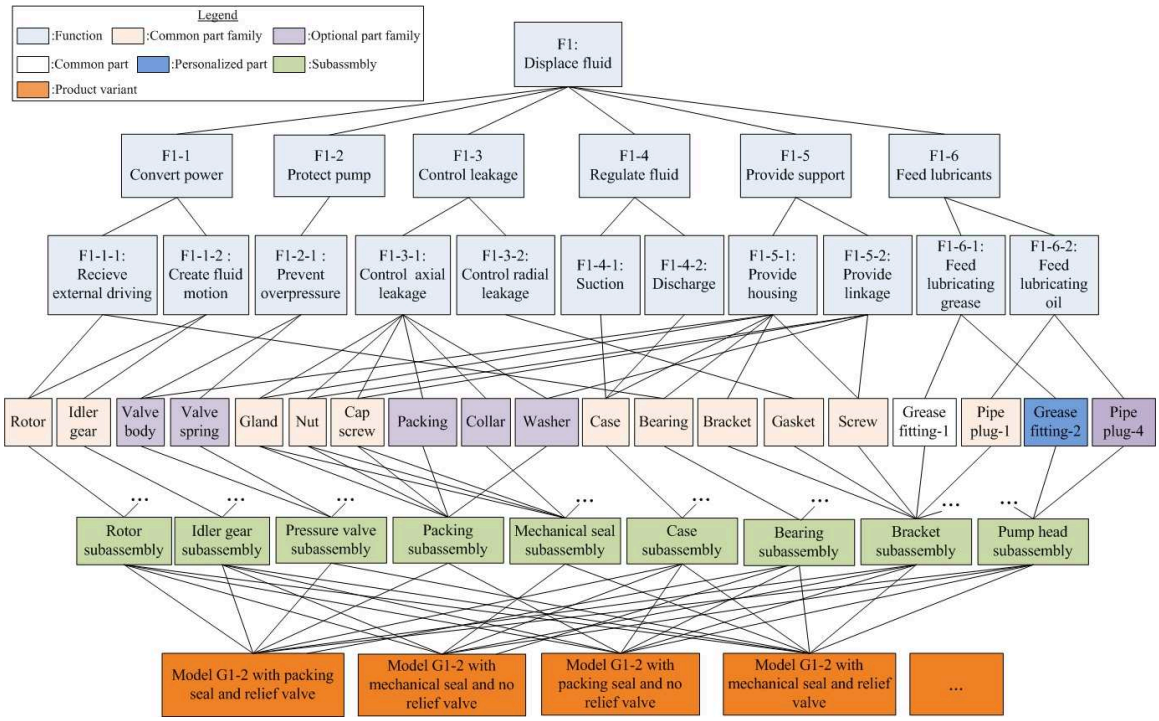


Figure 2-3: Mapping between function elements and product structure in PVDA illustrated by a gear pump family example

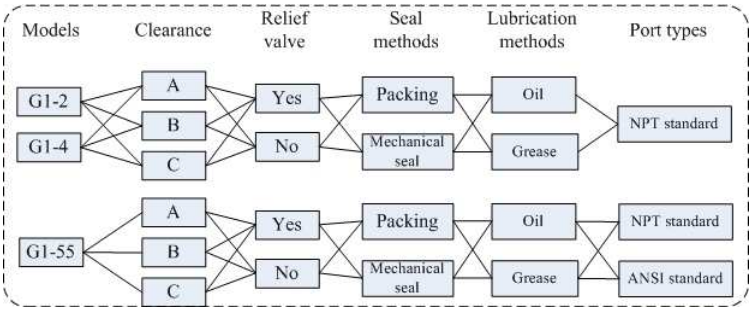


Figure 2-4: Attributes and attribute values of a gear pump family

to the bushing inside the pump head is a personalized parts because it delivers the exclusive functions in one specific product variant.

2.1.2 Proposition of logical operators to represent configuration relations

The representation in figure 2-3 is not enough in terms of all the necessary information on the product structure of a product family, because it is vague on the configuration relations between parts of product variants. For example, figure 2-3 can not explicitly represent the configuration relation between the parts for packing seal and the parts for mechanical seal,

which is an exclusive relation. In addition, the representation is not concise in terms of linkages between parts and higher levels of components and product variants: too many redundant lines between them. Therefore, we propose three types of logical operators to represent the configuration relations in the product structure of a product family in order to simplify the representation(Xia et al., 2015):

- **AND** operator: all physical components connected to an **AND** operator must appear in the same product variant;
- **XOR** operator: the physical components connected to a **XOR** operator are exclusive of each other, which means only one of them can appear in a product variant;
- **OPTION** operator: the physical component connected to an **OPTION** operator is optional for a product variant.

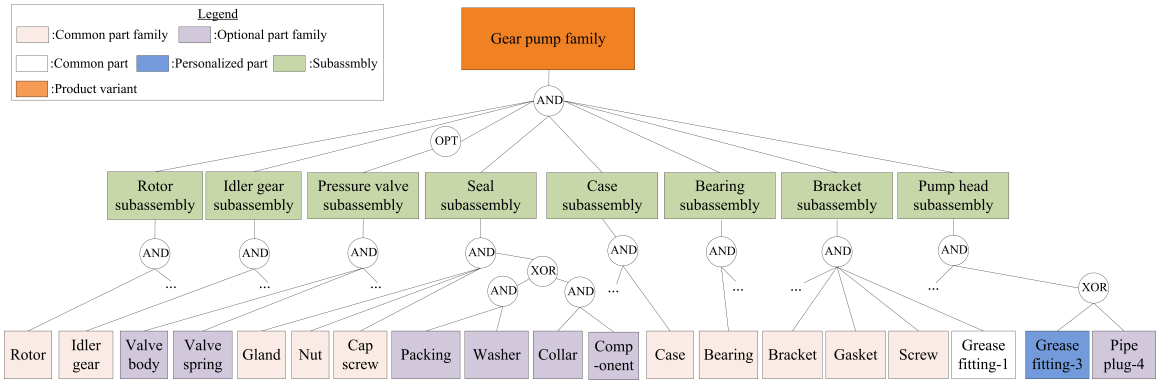


Figure 2-5: Partial product structure of a gear pump family organized by using the three logical operators

Figure 2-5 reorganizes the product structure described in figure 2-3 by using the three logical operators. The three logical operators can be combined to express more complex configuration relations. For example, packing and mechanical seal are two seal options for the pump variants, they share a set of common part families:{*gland*, *nut*, *cap screw*}, but each of them has optional part families: *packing* and *washer* for packing option, *collar* and *component seal* for mechanical seal option, as shown in figure 2-5. In order to represent the configuration relations between these part families, in figure 2-5 an **AND** operator is used to connect all the common part families with a **XOR** operator that expresses an exclusive relation between the two seal options, the **XOR** operator is connected with two additional

AND operators, each of which is connected with the corresponding part families for the two seal options.

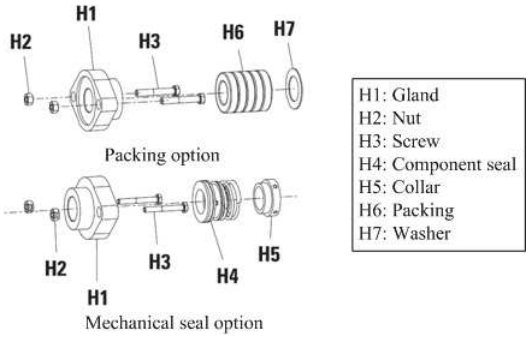


Figure 2-6: Two seal options for a gear pump family

2.1.3 Proposition of variety representation for part families

In the product architecture of product variety, the physical structure of the product variants are represented through organizing the parts of these product variants into four categories of parts (common parts, personalized parts, common part families and optional part families) and then connecting them with three types of logical operators according to their configuration relations in the product variants. Being different with common parts and personalized parts whose design specifications maintain the same in the product variants, part families consist of similar part variants whose design specifications change to some extant. Therefore, PVDA needs to be able to represent the variety and configuration of a part family.

As mentioned in the literature review, feature-based methods are an effective modeling paradigm for the representation of the design specifications of parts without involving geometric computation. In PVDA, we decompose the parts of a product variant into a set of features, each feature implements a part of the technical functions of the part. The part variants of a part family have some commonalities on their feature configurations. In order to take advantage of the commonalities between the part variants instead of representing every part variant individually, we propose a part variety decomposition network. In the variety decomposition network, part variety is divided into three correlative decomposition levels: Function module level, feature cluster level and feature variant level.

- *Function Module (FM) level*

  From a design point of view, a part is designed to be assembled with other parts to

become a functional component of a final product. Thus, the features of a part are designed to fulfill either technical functions or assembling functions. The technical functions delivered by the features could be the further decompositions from the function structure of the product family. Function modules at the FM level reflect the part variety on design functions delivered by the feature variants on part variants.

- *Feature Cluster (FC) level*

  Definition 9 (*Feature cluster*)**.** A feature cluster refers to a group of similar features belonging to the same feature type, and in the meantime serving the same design functions on different part variants.

  The features in a feature cluster are called feature instances. All the feature instances inherit the same set of geometric attributes, but the values of those attributes could be different. Feature clusters on the FC level represent the part variety on feature types.

- *Feature Variant (FV) level*

  FV level is the bottom level of the part variety decomposition network. It consists of the feature instances in the feature clusters. Each feature instance is a modular information container which holds a part of design specifications of a part variant. It has an object-oriented structure in which the geometric and technical attributions of each feature instance can be instantiated from the corresponding form features predefined in feature taxonomy.

Similar to the mappings between function structure and physical components in the product architecture, the links between the elements of both FM and FC levels could have four types of mappings:

- One-to-one mapping. One function module is mapped to one feature cluster. This is a kind of modular mapping which means the feature cluster in this mapping has no functional coupling with other feature clusters.

- One-to-many mapping. One function module is mapped to multiple feature clusters. In this mapping, the feature clusters deliver a common function.

- Many-to-one mapping. Multiple function modules are mapped to one feature cluster. In this case, one feature cluster implements multiple function modules.

- Many-to-many mapping. In this case, the function modules couple with each other in terms of feature clusters, and feature clusters also couple with each other in terms of function modules.

The variety configurations for the feature variants in one feature cluster have two possible situations:

- The feature variants in the feature cluster belong to one particular part variant. This situation corresponds to the fact that it is possible to have multiple features on one part variant that belong to the same feature type and deliver the same functions;

- The feature variants in the feature cluster belong to different part variants. For different part variants, it is possible to apply the same type of feature to realize the same function module. Therefore, it is possible that for the feature variants in a feature cluster to be held by different part variants.

In variety decomposition network, the three types of logical operators are also applicable to describe the configuration relations among the components of a part family. Here, an oil pump body family is used to illustrate the part variety decomposition network. In this part family, there are two part variants as shown in figure 2-7. Figure 2-7 also shows a portion of design features of these two oil pump body variants. Table 2.1 lists the features shown in figure 2-7. According to the geometrical characteristics and manufacturing processes of these features, they can be classified into three feature categories: *pocket*, *hole* and *chamfer*. For oil pump body 1, it has 1 *pocket*, 2 *holes* and 1 *chamfer* in figure 2-7, while for oil pump 2, it has 4 *pockets*, 1 *hole*, and 2 *chamfers*. The two oil pump bodies have similar feature type configuration, but the number of feature variants and their geometrical dimensions and tolerances may vary.

An oil pump body is a sub-assembly of an oil pump. The main design purpose of an oil pump body is to provide steady and suitable support for the other sub-assemblies, like pump cover, driving shaft, driven shaft and gears. Each feature on the oil pump body is designed to serve one or multiple technical functions related to the main purpose of the oil pump body. Table 2.2 lists the design functions for the design features shown in figure 2-7.

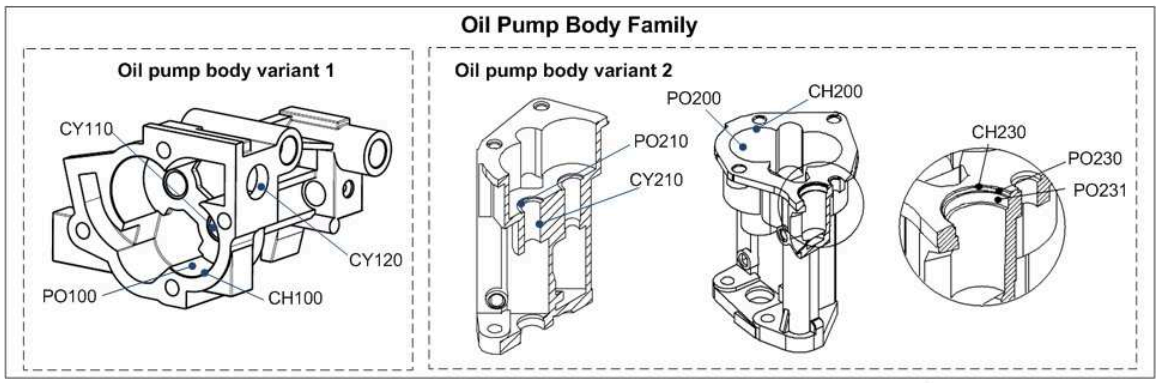


Figure 2-7: Oil pump body family and its partial design features

For oil pump body 1, its features implement *F1*, *F2* and *F3*, while oil pump body 2 has the features designed for *F1*, *F2* and *F4*.

Table 2.1: Feature information of an oil pump body family

Product variants	**Feature types**		
	Pocket	**Hole**	**Chamfer**
Oil pump body 1	PO100	CY110,CY120	CH100
Oil pump body 2	PO200,PO230,PO231,PO210	CY210	CH200,CH230

Table 2.2: Design functions of the design features on the two pump body variants

Design functions	**Design features**
Positioning the driving gear (F1)	PO100,CH100,PO200,CH200
Positioning the driving shaft (F2)	CY110,CY210,PO210
Positioning the oil outlet (F3)	CY120
Positioning the pressure valve (F4)	PO230,PO231,CH230

With the given information on this oil pump body family, the part variety decomposition network for this part family can be constructed as shown in figure 2-8. The FM level consists of the four functions, that F1, F2, F3, and F4. According to definition 9, the feature variants in this example can be grouped into 7 feature clusters, therefore, on the FC level of the part variety decomposition network, there are 7 feature clusters including *pocket cluster for F1 (PC1)*, *chamfer cluster for F1(CC1)*, *hole cluster for F2 (HC2)*, *pocket cluster for F2 (PC2)*, *hole cluster for F3 (HC3)*, *pocket cluster for F4 (PC4)* and *chamfer cluster for F4 (CC4)*. The feature variants of each feature cluster are organized on the FV level.

The three logical operators are used to represent the configuration relations among the

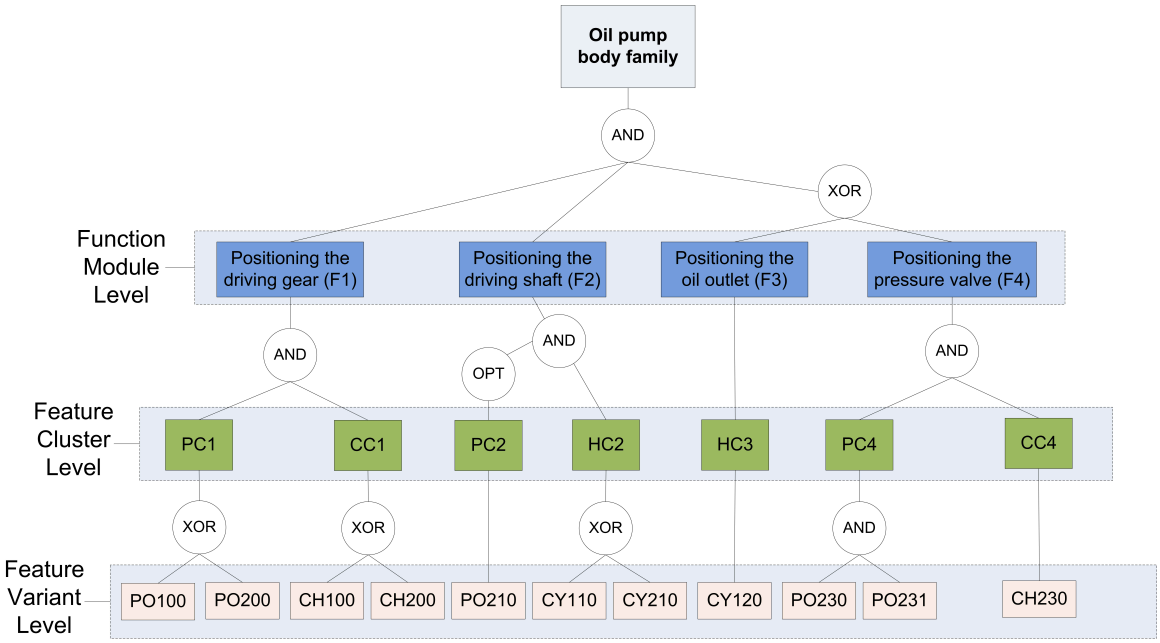


Figure 2-8: Part variety decomposition network for the oil pump body example

variety components linked to the same uper level component. An **AND** operator and a **XOR** operator are applied together to express the configuration relations among the four design functions on the function module level; *F1* and *F2* are the common functions of all the part variants, as they are connected to an **AND** operator; while the part variants must have either *F3* or *F4* because of the **XOR** operator. On the feature cluster level, an **AND** operator and an **OPTION** operator are used to describe the configuration relatinos between the feature clusters. The **AND** operator connecting *F1* with a pocket cluster and a chamfer cluster indicates that both of these feature clusters should be chosen to achieve *F1*; the **OPTION** operator and the **AND** operator connecting *F2* with a pocket cluster (*PC2*) and a hole cluster (*HC2*) means that *PC2* is optional for *F2*, but *HC2* is mandatory for *F2*. On feature variant level, the feature variants from the same feature cluster are linked by either **XOR** or **AND** operator. It should be noticed that there exist some variety components which are connected directly to their upper level component; in these cases, the upper level component is connected with a single lower level component which is not an optional component.

2.2 Proposition of product variety configuration constraints

2.2.1 A propositional-logic-based representation scheme

In PVDA, all the product variety components are organized into a network structure with three types of logical operators to represent the configuration relations. However, the configuration relations represented in PVDA are the ones among the child components of the same parent component from the upper decomposition level. For example, in the product variety architecture, the logical operators are used to represent the configuration relations among the part families whose parent component is the same product subassembly from subassembly level, and in part variety architecture, the logical operators are used to represent the configuration relations among the feature variants belonging to the same feature cluster.

However, PVDA has no mechanism to determine the configuration relations among the child components of different parent components from the upper decomposition level. For example, in the gear pump family example shown in figure 2-5, a specific *rotor* variant in *rotor* family requires a corresponding *idler gear* variant in *idler gear* family according to the model chosen for the pump variant; in the oil pump body family example shown in figure 2-8, the feature variant *PO100* in the feature cluster *PC1* requires a feature variant *CY110* in the feature cluster *HC2*.

In addition, the configuration constraints can either be among the configurable components in PVDA, or among the attributes' values of the product or the components of the product, or among certain attributes' values and certain configurable components. For example, in the gear pump family, if the attribute of a product variant: $model.product$ = "$G1-55$" and $valve_assembly.product$ = "Yes", then the attribute of the spring in the relief valve: $pressure.spring$ = "200". If the attribute of a product variant: $valveAssembly.product$ = "No", then a cover should be chosen as one component for this product variant.

Therefore, it is necessary to develop a mechanism which can be used to represent the configuration relations among the configurable components under the two situations described above. This subsection introduces a propositional-logic-based scheme to give a universal representation of the variety configuration constraints between the variety components in PVDA (Xia et al., 2016).

Definition 10. Let α_P be a propositional formula and let $P = \{p_1, p_2, ..., p_n\}$ be a set of

atomic propositions appearing in α_P, if $P = \phi$, then α_P is valid.

Definition 11. A configuration constraint has the form: $\alpha_{P_1} \leftrightarrow \alpha_{P_2}$ where α_{P_1} and α_{P_2} are two propositional formulas and each atomic proposition in P_1 and P_2 corresponds to a variety component in the variety decomposition network.

Definition 12. Let $\varphi_{(\alpha_{P_1} \leftrightarrow \alpha_{P_2})}$ be an interpretation for $\alpha_{P_1} \leftrightarrow \alpha_{P_2}$ which assigns one of the truth values (True:1 or False:0) to every atomic proposition in P_1 and P_2.

Definition 13. An interpretation $\varphi_{(\alpha_{P_1} \leftrightarrow \alpha_{P_2})}$ is a variety configuration for the variety components in a configuration constraint $\alpha_{P_1} \leftrightarrow \alpha_{P_2}$, if and only if it satisfies $\alpha_{P_1} \wedge \alpha_{P_2}$, that is the truth value of $\alpha_{P_1} \wedge \alpha_{P_2}$ under $\varphi_{(\alpha_{P_1} \leftrightarrow \alpha_{P_2})}$ is True.

2.2.2 Illustration examples

In PVDA, the configurable components come from two variety levels, product family level and part family level. For product family level, the configurable components consist of function modules, functional sub-assemblies, part families and personalized parts; the common parts in the product variety architecture are unconfigurable components because they are obligatory for every product variant in the family. For part family level, the configurable components include function modules, feature clusters and feature variants in the part variety decomposition network.

As described in previous subsection, there are two situations for the configuration constraints between the configurable components in PVDA:

- Configuration Situation I: the configuration relations between the child components of the same upper level component;
- Configuration Situation II: the configuration relations between the child components of the different upper level components.

In situation I, the three logical operators are used to represent the configuration relations in a visualized way. As the three logical operators can be expressed by logical formulas, the configuration relations in this situation can be expressed by the propositional-logic-based scheme with the help of the logical formulas for the three logical operators. For example, in the product structure of the gear pump family shown in figure 2-5, the configuration

constraint on the seven configurable part families for the seal subassembly, {gland, nut, cap screw, packing, washer, collar}, can be expressed as the following formula:

$$\begin{aligned} P_{seal} \quad \leftrightarrow \quad & P_{gland} \wedge P_{nut} \wedge P_{capScrew} \wedge \\ & ((P_{packing} \wedge P_{washer} \wedge \neg P_{component} \wedge \neg P_{collar}) \vee \\ & (\neg P_{packing} \wedge \neg P_{washer} \wedge P_{component} \wedge P_{collar})). \end{aligned}$$

Similarly, in the part variety decomposition network for the oil pump body family shown in figure 2-8, the configuration constraint on the two feature clusters for function module 2, $\{PC2, HC2\}$, can be expressed as the following formula:

$$P_{F2} \leftrightarrow P_{HC2} \wedge (P_{PC2} \vee \neg P_{PC2}).$$

In situation II, the propositional-logic-based scheme is used directly for the configuration constraints on the configurable components. For example, in the product family example, a specific rotor variant requires a specific idler gear variant. This configuration constraint can be described as the following formula:

$$P_{rotor1} \leftrightarrow P_{idlerGear1}.$$

In the part family example, the configuration constraint on the feature clusters, PC2, CC4 and PC4 can be described as the following formula:

$$P_{PC2} \leftrightarrow P_{CC4} \wedge P_{PC4}.$$

Instead of the configuration constraints on the configurable components of the product/part family, the scheme can also be used to express the configuration constraints on the attributes of the configurable components or of the product/part family. For example, in the product family example, there exist configuration constraints on the model and the clearance of the gear pump variant, the viscosity and the temperature of the fluid pumped in the

variant. One of these configuration constraints can be described as the following formula:

$$\begin{aligned} P_{clearance=\text{"A"}} \quad \leftrightarrow \quad & ((P_{viscosity \leq 540} \wedge P_{model=\text{"G1-2"}}) \vee \\ & (P_{viscosity \leq 160} \wedge (P_{model=\text{"G1-4"}} \vee P_{model=\text{"G1-55"}}))) \wedge \\ & P_{temperature \leq 107}. \end{aligned}$$

The scheme can also be used to express the configuration relations between the attributes and the configurable components. For example, the gear pump variant in model G1-4 or model G1-55 requires semi-round rings in its bearing subassembly, while the one in model G1-2 does not. This configuration constraint can be described as the following formula:

$$P_{model=\text{"G1-4"}} \vee P_{model=\text{"G1-55"}} \leftrightarrow P_{semiRoundRing}.$$

$$P_{model=\text{"G1-2"}} \leftrightarrow \neg P_{semiRoundRing}.$$

2.3 Propositions of process-related information representation

Product variety decomposition architecture and propositional-logic-based representation scheme represent the configuration-related information of a product family. Beside the configuration-related information, product variety model also needs to accommodate the process-related design information in order to support the automatic generation of reconfigurable process plans for a product/part family. At product level, the variety model should be capable to represent the design specifications for assembly process planning; at part level, the model should be able to capture the necessary information for machining process planning. In addition, since a product/part family consists of a group of variants, the model should handle the commonalities of the process-related information among these variants so as to achieve lower data redundancy and higher processing efficiency.

2.3.1 Representation of process-related information for part variety

Part variety decomposition network splits part variety into three levels, function module level, feature cluster level and feature variant level. Function module level represents the function variety of part variety, while feature cluster level and feature variant level are used to capture the variety of design specifications of a part family. Therefore, the process-related

information of a part family come from feature cluster level and feature variant level in the part variety decomposition network.

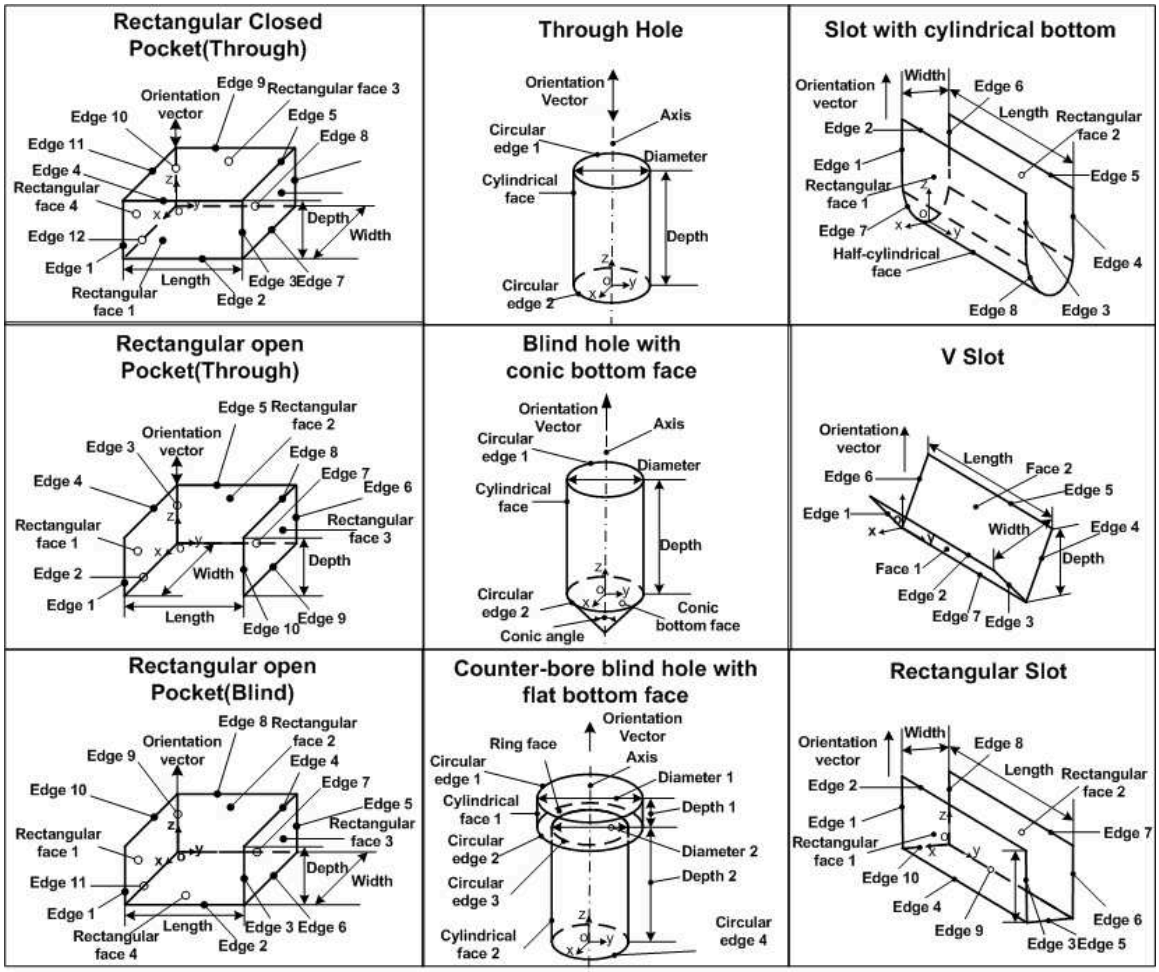


Figure 2-9: Several predefined form features

Feature-based modeling provides an object-oriented method to represent design specifications of mechanical parts. In this method, each feature is a modular entity instantiated from a predefined form feature. A form feature represents a class of features which have the same geometric structure and attribute structure. Feature taxonomy represents the classification and relationships between all the predefined form features.

Each form feature defines the geometric specifications of a class of features, including the geometric elements in the form feature, the orientation as well as the shapes, the topological relations and the attributes of these geometric elements. Figure 2-9 shows several predefined form features (Catania, 1991; Gu, 1994; Gonzalez and Rosado, 2004).

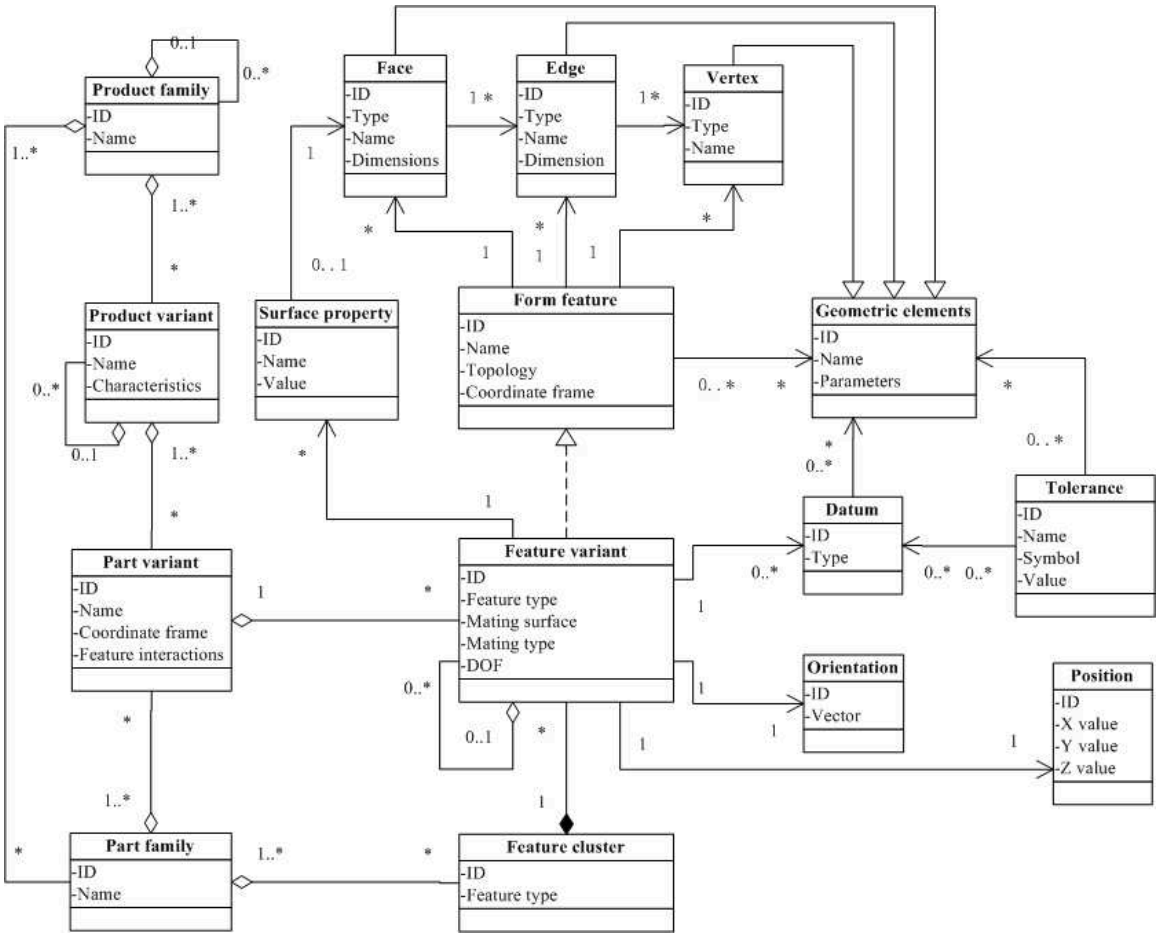


Figure 2-10: UML class diagram of feature-based product variety model

Figure 2-10 shows a UML class diagram of the feature-based product variety model. The feature variants on parts are the instantiations of the corresponding form features. Thus, the geometric elements as well as their shapes, topological relations and attributes in the feature variants inherit from the form features, while the attribute values of the geometric elements could be different among the feature variants instantiated from the same form feature. Besides the geometric attributes inherited from form features, the process-related information for feature variants in parts also includes the functional attributes not defined in the form features:

- Position. This attribute specifies the position of a feature variant in the part's coordinate frame;

- Orientation. This attribute indicates the orientation of a feature variant in the part's

coordinate frame;

- Surface properties. These attributes determine the surface roughness and hardness of the important surface of a feature variant;
- Tolerances. These attributes define the dimensional tolerances and geometrical tolerances of the geometric elements of a feature variant;
- Mating attributes. Only the feature variants used as assembly features have these attributes. They consists of mating surfaces, mating type, and Degree Of Freedom (DOF);
- Datum. These attributes mark the geometric elements of a feature variant acting as datum for dimensioning or tolerancing.

The orientations and position of feature variants in the coordinate frame of the parts can be calculated by using a 4 x 4 transform matrix. The mathematical form of this calculation is

$$\begin{bmatrix} \overrightarrow{P'} \\ 1 \end{bmatrix} = \begin{bmatrix} R & \overrightarrow{Tr} \\ \overrightarrow{\mathbf{0}^T} & 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{P} \\ 1 \end{bmatrix}; \tag{2.1}$$

$$\overrightarrow{Or} = \overrightarrow{P'} - \overrightarrow{Tr}; \tag{2.2}$$

$$\overrightarrow{Po} = \overrightarrow{Tr}; \tag{2.3}$$

In formula (2-4), $\overrightarrow{P'}$ is the vector related to the orientation of the feature variant in the coordinate frame of a part, $\overrightarrow{P}$ is the original orientation vector defined in the form feature. R is a 3 x 3 rotation matrix representing the rotation of the coordinate frame of the feature variant relative to the coordinate frame of the part. $\overrightarrow{Tr}$ is a translation vector indicating the translation of the feature variant's coordinate frame relative to the part's coordinate frame. $\overrightarrow{\mathbf{0}}$ is a zero vector. The orientation of the feature variant $\overrightarrow{Or}$ in the part's coordinate frame equals $\overrightarrow{P'} - \overrightarrow{Tr}$, and the position of the feature variant $\overrightarrow{Po}$ in the part's coordinate frame equals the translation vector $\overrightarrow{Tr}$.

The rotation matrix R can be further decomposed into three elements, $R_{x,\theta}$, $R_{y,\alpha}$, $R_{z,\omega}$. $R_{x,\theta}$ indicates rotation by an angle θ about the x axis of the part's coordinate frame. $R_{y,\alpha}$

indicates rotation by an angle α about the y axis of the part's coordinate frame. $R_{z,\omega}$ indicates rotation by an angle ω about the z axis of the part's coordinate frame. R is derived by multiplying these three elemental rotation matrix sequentially. $R_{x,\theta}$, $R_{y,\alpha}$, $R_{z,\omega}$ have the following mathematical form:

$$\left[R_{x,\theta}\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}, \theta > 0 \text{ when clockwise rotation, else } \theta \leq 0; \tag{2.4}$$

$$\left[R_{y,\alpha}\right] = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}, \alpha > 0 \text{ when clockwise rotation, else } \alpha \leq 0; \tag{2.5}$$

$$\left[R_{z,\omega}\right] = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}, \omega > 0 \text{ when clockwise rotation, else } \omega \leq 0; \tag{2.6}$$

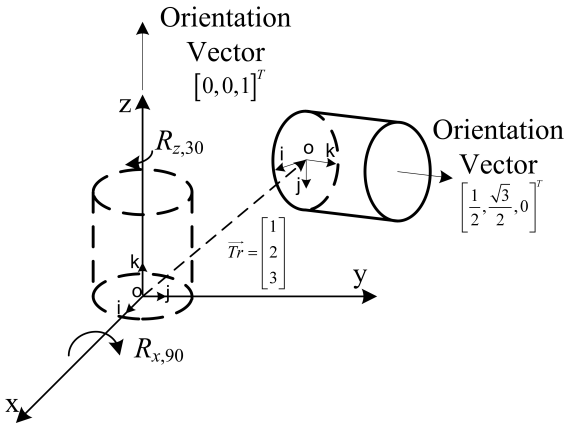


Figure 2-11: Orientation of a hole variant in a part's coordinate system

For example, in figure 2-11, a hole feature variant is located in a part's coordinate frame x-y-z with two rotations, $R_{x,90}$, $R_{z,30}$ and one translation, $\overrightarrow{Tr} = [1 \quad 2 \quad 3]^T$, and the original orientation vector defined in the form feature is $[0 \quad 0 \quad 1]^T$, then R in the 4 x 4 transform matrix for the original orientation vector can be calculated by using the formulas (2-7) and

(2-9):

$$R = R_{z,30}R_{x,90} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & -1 & 0 \end{bmatrix}.$$

The orientations and position of feature variants on parts can be calculated by applying the formulas (2-4), (2-5) and (2-6):

$$\begin{bmatrix} \overrightarrow{P'} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} & 1 \\ -\frac{1}{2} & 0 & \frac{\sqrt{3}}{2} & 2 \\ 0 & -1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{4+\sqrt{3}}{2} \\ 3 \\ 1 \end{bmatrix};$$

$$\overrightarrow{Or} = [\frac{1}{2} \quad \frac{\sqrt{3}}{2} \quad 0]^T;$$

$$\overrightarrow{Po} = [1 \quad 2 \quad 3]^T.$$

In PVDA, a feature cluster is a collection of similar feature variants instantiated from the same form feature but varying on the attribute values. The description for a feature cluster is the union of the description for the feature variants in this feature cluster. Two data types are adopted to express the attribute values of a feature cluster:

- Interval, for the attributes which take continuous values;
- Set, for the attributes which take discrete values.

The attribute values of the same attribute of the feature variants are gathered into an interval or a set so as to form the attribute value for the corresponding attribute of the feature cluster where the feature variants belong. Figure 2-12 shows a part of the attribute values of a feature cluster: $HC2$ for the oil pump body family. Intervals and sets are used to describe the attribute values of this feature cluster. The same attribute values of different feature variants are only considered once for the corresponding attribute value of feature cluster, like the position and the orientation; an attribute value represented by an interval or a set means that all of the feature variants in this feature cluster have a design specification

on this attribute whose value is limited in this interval or set.

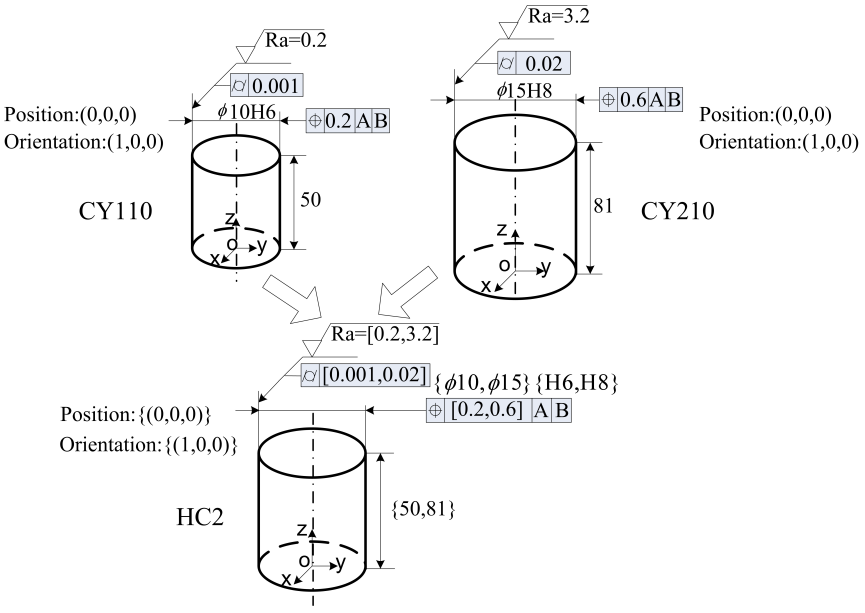


Figure 2-12: Attribute values of the feature cluster: $HC2$ for the oil pump body family

2.3.2 Representation of the interactions between the variety components in PVDA

Feature interaction

In general, two situations of feature interactions are considered in this thesis:

- The first one is the interactions between the feature variants in one part variant;
- The second one is the interactions between the feature variants in the mating part variants in an assembly.

Feature interactions have great influences on the manufacturing process sequence for part variants and product variants. In the first situation, feature interactions impact the machining operation sequence for the involved feature variants. For example, if there is a thread resides on a cylinder, then the cylinder must be machined to specifications before the thread is cut on it; if two holes share the same axis, then the smaller hole should be drilled prior to the bigger hole so that the driller while machining the bigger hole experiences less stress. In the second situation, feature interactions affect the assembly sequence of the involved part variants. For example, in figure 2-13, the feature interaction between the

planes of part A and part C causes the impossibility to realize the feature interactions between the planes of part C and part B and between the cylinders of part C and part B, thus, the feature interactions between B and C have to be realized before that between A and B.

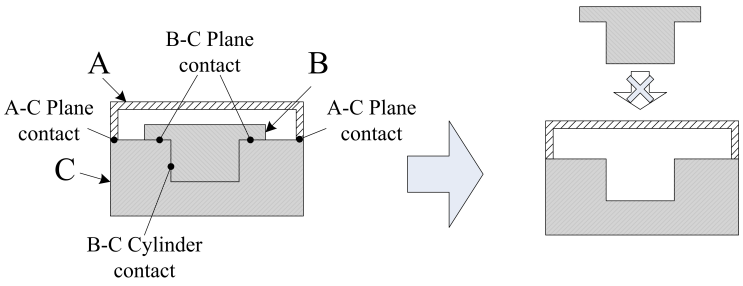


Figure 2-13: Illustration for the influence of feature interaction on assembly sequence

In the first situation of feature interactions, there exist two cases of interaction between feature variants:

- Tolerance/datum dependencies. In this case, the tolerance specifications establish one or some feature(s) as the datum feature(s), and other features are constrained by the tolerance in relation to the datum feature. Figure 2-14 shows a case for tolerance/datum dependencies. Hole 1 and hole 2 have feature interaction because hole 2 is used as a datum for hole 1 in a parallelism specification.

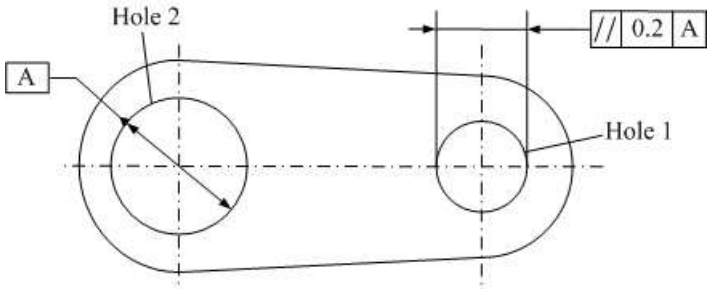


Figure 2-14: Illustration for tolerance and datum dependencies

- Topological interactions. In this case, there is either a distance relationship between features or a volumetric intersection between features. Figure 2-15 shows a case for distance relationships between features. In the figure, shaft hole 1 is used as a distance reference for shaft hole 2, tapped hole 1, tapped hole 2, tapped hole 3 and tapped hole 4. Figure 2-16 illustrates a case for volumetric interactions between hole features.

Eight types of volumetric intersections between hole features are identified by Etienne et al. (2006). The principles of classifications can be extended to the volumetric intersections between other common features like slot, pocket and solid, as illustrated in figure 2-17.

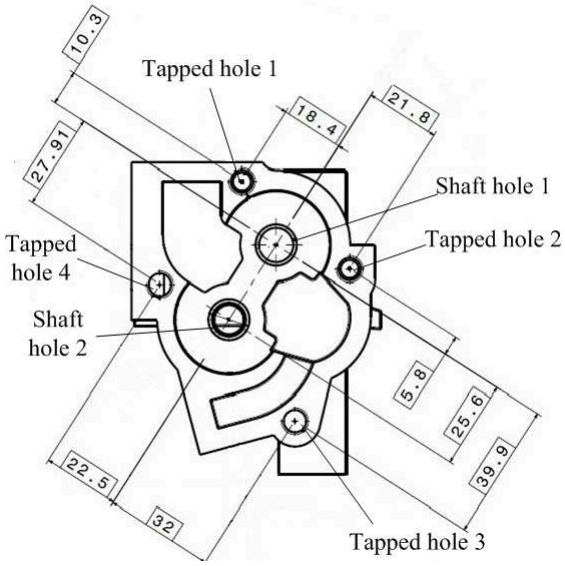


Figure 2-15: Illustration for the distance relationship between features

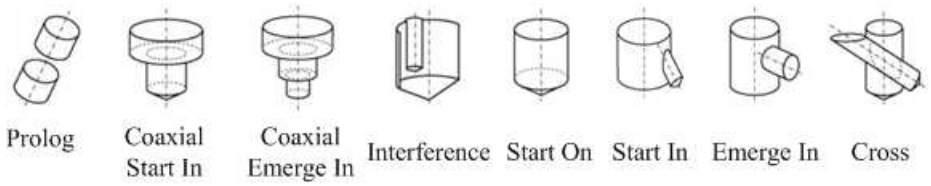


Figure 2-16: Illustration for volumetric interaction between hole features (Etienne et al., 2006)

A knowledge-based representation approach is used to model the feature interaction between the feature variants in one part variant in a computer interpretable way. In this approach, unary predicates represent the basic types of objects in the knowledge domain. For example: $Hole(x)$, $Pocket(y)$, $Plane(z)$, $Slot(l)$ represent the feature instances instantiated from different form features; $Axis$(a), $Surface$(s), $Edge$(e) indicate the geometric element objects in the feature instances.

N-ary predicates express the relationships among the objects. Those relationships could be the attribute relationships, tolerance/datum dependencies, and topological interactions. For example:

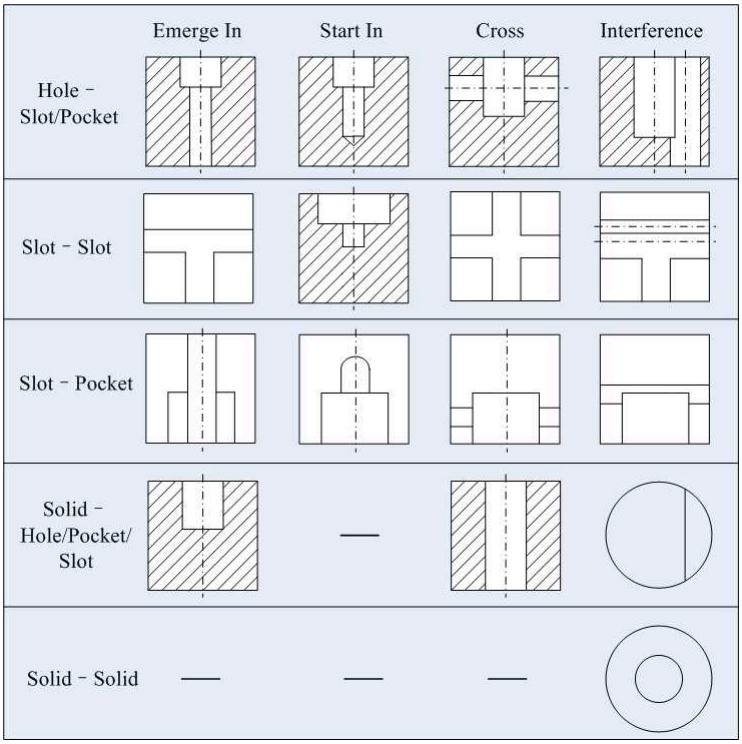


Figure 2-17: Volumetric interactions between hole, slot, pocket, solid and plane

- $hasAxis(Hole(\mathrm{x}), Axis(\mathrm{a}))$ represents that Hole x has an axis a;
- $hasRoughness(Surface(\mathrm{s}), \mathrm{ra}, 3.2)$ expresses that a surface s has a surface roughness specification Ra = 3.2;
- $hasPerpendicularity(Axis(\mathrm{a}), Datum(Plane(\mathrm{p})), \phi 0.01)$ means there is a perpendicularity on axis a that restricts its variations within a cylindrical zone of diameter 0.01, which is perpendicular to datum plane p.
- $hasDistance(Surface(\mathrm{s1}), Surface(\mathrm{s2}), 5, Vector(0,0,1))$ indicates that the distance between surface $s1$ and surface $s2$ in the direction of a vector $(0,0,1)$ is 5mm.
- $emergeIn(Hole(\mathrm{x}), Hole(\mathrm{y}))$ denotes that hole x has a volumetric interaction with hole-y where hole y emerges in hole x from the inside to the outside.

The parallelism tolerance specification in figure 2-14 can be represented by using the following predicates:

$$\{hasParallelism(\mathrm{a}, \mathrm{b}, 0.2), Hole(\mathrm{hole1}), Hole(\mathrm{hole2}),$$
$$hasAxis(\mathrm{hole1}, \mathrm{a}), hasAxis(\mathrm{hole2}, \mathrm{b}), Axis(\mathrm{a}), Axis(\mathrm{b})\}.$$

The distances between tapped hole 1(th1) and shaft hole 1(sh1) in figure 2-15 can be represented by using the following predicates:

$$\{hasDistance(\mathrm{d}, \mathrm{e}, 18.4, Vector(1,0,0)), hasDistance(\mathrm{d}, \mathrm{e}, 10.3, Vector(0,1,0)),$$
$$Hole(\mathrm{sh1}), Hole(\mathrm{th1}), hasAxis(\mathrm{sh1}, \mathrm{d}), hasAxis(\mathrm{th1}, \mathrm{e}), Axis(\mathrm{d}), Axis(\mathrm{e})\}.$$

When considering the feature interactions at the level of part family, the situation becomes complicated. A feature interaction could maintain the same between all the FVs inside the involved FC. It is also possible for the FCs whose FVs involved in different feature interactions. Therefore, in order to maximize the benefits of commonality, in feature-based part variety model, the collective feature interactions of a part family are represented at FC level, while the diversity of feature interactions is kept at FV level. A rule is defined to determine a feature interaction between two FCs at FC level(Xia et al., 2016):

Assume that:

1. FC_1=$\{\mathrm{fv}_1, \mathrm{fv}_1, ..., \mathrm{fv}_n\}$ is a feature cluster whose feature variant is $\mathrm{fv}_{k\in[1,n]}$;
2. FC_2=$\{\mathrm{fv}'_1, \mathrm{fv}'_1, ..., \mathrm{fv}'_m\}$ is a feature cluster whose feature variant is $\mathrm{fv}'_{k\in[1,m]}$;
3. FI=$\{\mathrm{fi}_1, \mathrm{fi}_2, ..., \mathrm{fi}_s\}$ is the set of feature interactions existing between the feature variants in FC_1 and the feature variants in FC_2;
4. $f(fi, FC)$ is a function that returns a feature variant $\in FC$ which involves in the feature interaction fi;
5. T=$\{\mathrm{T}_1, \mathrm{T}_2, ..., \mathrm{T}_r\}$ where $\mathrm{T}_{k\in[1,r]}$ is a set of feature interactions in FI having the same type. $\mathrm{T}_k \subseteq$ FI and $\bigcup_1^r T_k$=FI.

Rule 1. If $\exists \mathrm{T}_i$=$\{\mathrm{fi}'_1, \mathrm{fi}'_2, ..., \mathrm{fi}'_u\}$∈T such that $\bigcup_1^u f(\mathrm{fi}'_k, \mathrm{FC}_1)$=$\mathrm{FC}_1$ and $\bigcup_1^u f(\mathrm{fi}'_k, \mathrm{FC}_2)$=$\mathrm{FC}_2$, then T_i can be represented as a feature interaction between FC_1 and FC_2, the type of this feature interaction is the type of the feature interactions in T_i and the value of this feature interaction is a domain which contains all the values of the feature interactions in T_i.

Part interaction

In a product, part interactions happen between the parts having the assembly connections with each other. The representation of the part interactions in a product is important for

determining the assembly process plans for the product. The information that needs to be represented for part interactions includes:

- The relative position between the mating parts;
- The mating relations among all the parts in a product;
- The kinematic constraint information of each part in a product.

All the position of the parts can be explicitly represented by using 4 x 4 transform matrices with respect to a common coordinate frame, like the approach shown in subsection 2.3.1. Requicha and Whalen (1991) identify two drawbacks of this kind of explicit specification:

- Assembly representations typically are constructed by human designers, and it is difficult to define explicitly the required transforms;
- A specific transform defines a single point in the position space. Therefore, the method cannot describe the assemblies with moving parts.

A better approach is to represent the relative positions between the mating parts. As feature entities are used as a modular way to represent a part, the mating parts must have mating features on them. Therefore, the relative position between two mating parts can be easily represented through the relative position between their mating features. The mathematical form of this representation approach is:

$$T_{A-B} = T_{A-F_A} T_{F_A-F_B} T^{-1}_{B-F_B} \tag{2.7}$$

where T_{A-B} is a transform of part B's coordinate frame with respect to part A's coordinate frame, T_{A-F_A} is a transform of the mating feature F_A on part A relative to part A's coordinate frame, $T_{F_A-F_B}$ is a transform of the mating feature F_B on part B with respect to the mating feature F_A's coordinate frame, $T^{-1}_{B-F_B}$ is the inverse of the transform T_{B-F_B} that locates the mating feature F_B with respect to part B's coordinate frame. Figure 2-18 illustrates the transform from part A's frame to part B's frame via the mating feature's frames.

For the mating relations among all the parts, traditional liaison diagram can only show which parts are connected to each other. However, liaison diagram is a lightweight and concise method for modeling the mating relations among parts, which has been widely adopted

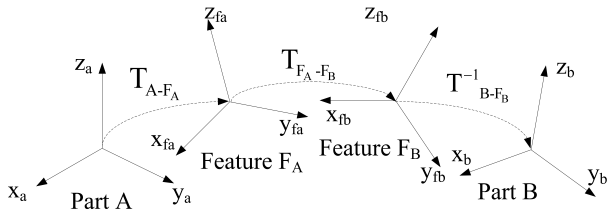


Figure 2-18: Relative position between two mating parts modeled by a transform chain

in assembly modeling. Besides liaison diagram, Whitney (2004) proposes a representation model named Datum Flow Chain (DFC) which has the capability to represent more information than a liaison diagram can do. A DFC model can accommodate the following information:

- Which parts are connected to each other;
- How the parts are connected to each other;
- How one part is located with respect to another part;
- Which DOFs of one part are constrained by another part;
- Which features are involved in the mating relations and which mating types (mating or contact) they belong to;
- Which key characteristics are realized;
- Which tolerances are involved in the mating relations.

Figure 2-19 shows a DFC model of a bearing housing assembly. A DFC is a graph-based representation. The big circles denote the parts of this bearing housing, the small black circles represent the mating feature involved into the assembly, the directed arcs means the part at its head is located with respect to the part at its tail. The directed arcs can be labeled to show which degrees of freedom it constrains. The sum of the DOFs constrained by all the incoming arcs to a part in a DFC should be equal to six unless there are some kinematic properties or other exceptional case. Each directed arc has an associated 4 x 4 transformation matrix that represents the position of the part at its head relative to the part at its tail (no shown in the figure 2-19). Two types of joints between the parts are distinguished. The parts connected by the directed arcs have mate as their joint type, which represent there are constraints and dimensional relationships established between these parts; while the joint

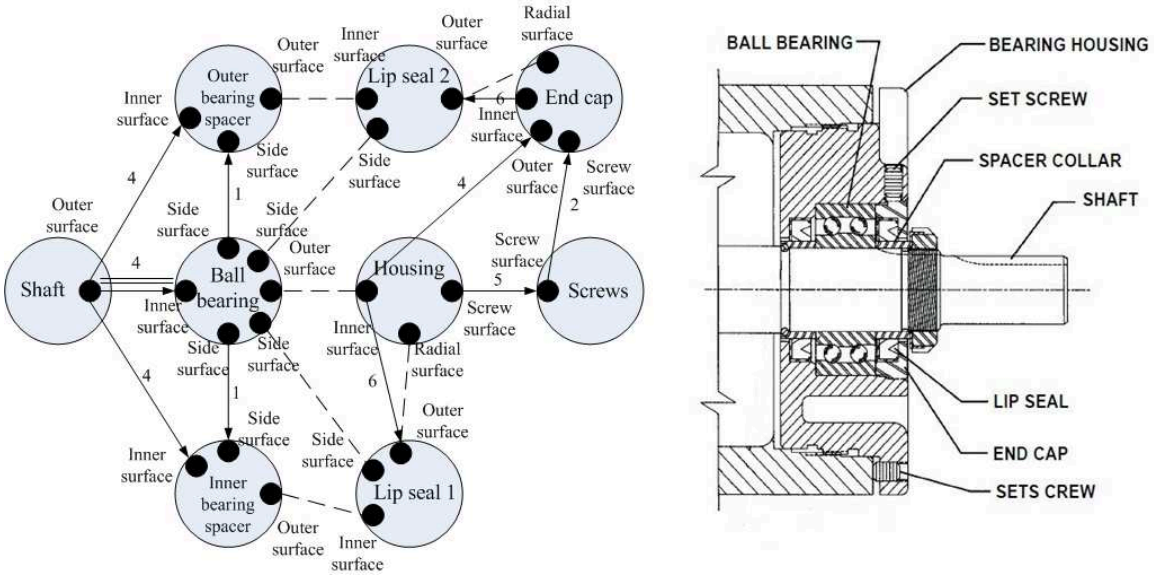


Figure 2-19: DFC model of a bearing housing assembly in a gear pump product variant

type between the parts connected by the undirected dashes is considered as contact, which means their joints merely support and fasten the part once it is located. Moreover, the key characteristics delivered by the parts are highlighted by double lines closed to the associated arcs.

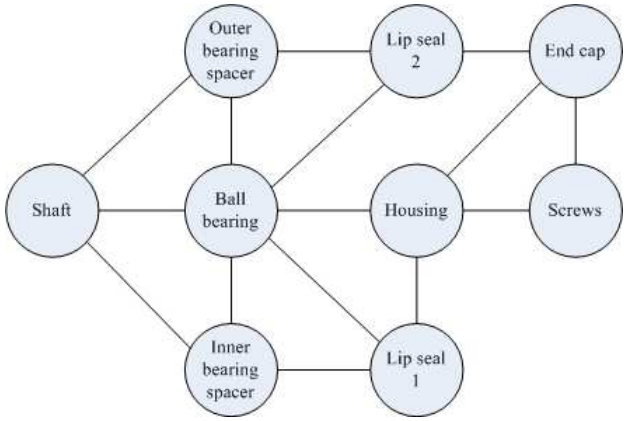


Figure 2-20: Liaison graph of the bearing housing assembly

Because DFC can establish all the connection constrains among the parts of an assembly, even at the level of parts' features, it can be considered as a detailed liaison graph. Given a DFC model, a liaison graph can be easily generated. In this thesis, DFC approach is used to represent the detailed mating information among all the part variants of one product variant. When only simple mating relations among the part variants are needed, a liaison

graph is generated.

In terms of a product family which contains a set of product variants, there could be the same mating relations among the mating parts of different product variants. If a detailed DFC model was defined to represent the mating information inside each product variant, those same mating relations would be repeatedly represented several times, which could cause data redundancy. Therefore, a representation mechanism should be developed to avoid repeatedly representing the common mating information among the mating parts of different product variants.

We propose a representation mechanism which represents the mating information between the mating parts at two levels:

- Product family level. The mating information represented at this level is shared by all the product variants.
- Product variant level. The mating information represented at this level is linked to particular product variants.

As mentioned in subsection 2.1.1, in PVDA, the physical variety components of a product family are classified into four types: Common Part (CP), part variants in a Common Part Family (CPF), part variants in an Optional Part Family (OPF)) and Personalized Part (PP). Considering the part interactions between the parts of these four types, there are 10 mating situations as shown in figure 2-23. Not all these mating situations can be represented at product family level. In terms of CPF and OPF, if not all of their part variants are involved in the same mating relation, the mating relations cannot be represented at product family level. For example, the mating relations between the part variants in CPF and a PP cannot be represented at product family level, because not all part variants in CPF have the mating relation with the PP. Table 2.3 lists that the mating situations in which the mating relations can be represented at product family level.

Based on this, we define the following fiction liaisons to represent the mating relations between the physical variety components at product family level according to the 10 mating situations:

Definition 14. $L_{CP-CP}(cp_1, cp_2) \in CP \times CP$, is a fictitious joint between a two common parts, cp_1 and cp_2, representing the mating relations between cp_1 and cp_2 at product family level.

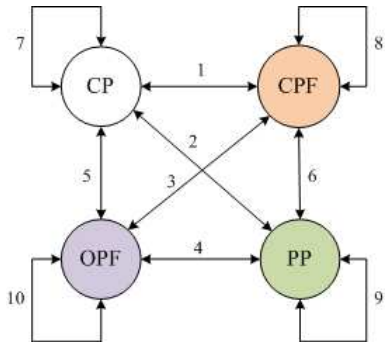


Figure 2-21: 10 mating situations between the physical variety components of a product

Table 2.3: Mating situations that can be represented at product family level

Types	CP	CPF	OPF	PP
CP	Yes	Yes	No	Yes
CPF	Yes	Yes	No	No
OPF	No	No	Yes	Yes
PP	Yes	No	Yes	Yes

Yes: it can be represented at product family level; No: it cannot be represented at product family level.

Definition 15. $L_{CP-PP}(cp_1, pp_1) \in CP \times PP$, is a fictitious joint between a common part, cp_1 and a personalized part, pp_1, representing the mating relations between cp_1 and pp_1 at product family level.

Definition 16. $L_{CPF-CP}(CPF_1, cp_1) \in CPF \times CP$, is a fictitious joint between a common part family, CPF_1, and a common part, cp_1, representing the mating relations between CPF_1 and cp_1 at product family level.

Definition 17. $L_{CPF-CPF}(CPF_1, CPF_2) \in CPF \times CPF$, is a fictitious joint between a common part family, CPF_1, and another common part family, CPF_2, representing the mating relations between CPF_1 and CPF_2 at product family level.

Definition 18. $L_{OPF-OPF}(OPF_1, OPF_2) \in OPF \times OPF$, is a fictitious joint between an optional part family, OPF_1, and an optional part family, OPF_2, representing the mating relations between OPF_1 and OPF_2 at product family level.

Definition 19. $L_{OPF-PP}(OPF_1, pp_1) \in OPF \times PP$, is a fictitious joint between an optional part family, OPF_1, and a personalized part, pp_1, representing the mating relations between OPF_1 and pp_1 at product family level.

Definition 20. $L_{PP-PP}(pp_1, pp_2) \in PP \times PP$, is a fictitious joint between two personalized parts, pp_1 and pp_2, representing the mating relations between pp_1 and pp_2 at product family level.

In terms of a part family, if it is involved in a mating relation at the level of product family, the mating relations between the mating features on these mating part variants are the same. Therefore, the mating relations between the mating features can be represented at the level of feature clusters where these mating features belong. A definition for this is given as follow:

Definition 21. $L_{FC-FC}(FC_1, FC_2) \in FC \times FC$, is a fictitious link between a feature cluster, FC_1 and another feature cluster, FC_2, defined by the set of liaisons between the feature variants of FC_1 and the feature variants of FC_2, so that $L_{FC-FC}(FC_1, FC_2)$ exists if and only if $\forall L_i(fv_n, fv_m) \in L_{FC-FC}(FC_1, FC_2) \wedge \forall L_j(fv_g, fv_k) \in L_{FC-FC}(FC_1, FC_2) \wedge (i \neq j) \wedge (fv_n, fv_g \in FC_1) \wedge (fv_m, fv_k \in FC_2) \wedge (L_i(fv_n, fv_m) = L_j(fv_g, fv_k))$.

According to the definition above, we define the following rules to determine the privileged mating information which can be represented at product family level in the 8 mating situations:

Rule 2. If and only if the mating relations between two common parts, cp_1 and cp_2, in all the product variants remain the same, then there exist a joint, $L_{CP-CP}(cp_1, cp_2)$ at product family level.

Rule 3. If and only if the mating relations between a common part, cp_1 and a personalized part, pp_1 in all the product variants involved by the mating relations remain the same, then there exist a joint, $L_{CP-PP}(cp_1, pp_1)$, at product family level.

Rule 4. If and only if the mating relations between two personalized parts, pp_1 and pp_2, in all the product variants involved by the mating relations remain the same, then there exist a joint, $L_{PP-PP}(pp_1, pp_2)$, at product family level.

Rule 5. If and only if all the part variants in a common part family, CPF_1, have the same mating relation with a common part, cp_1, then there exist a joint, $L_{CPF-CP}(CPF_1, cp_1)$, at the product family's level.

Rule 6. If and only if all the part variants in a common part family, CPF_1, have the same mating relation with the part variants in another common part family, CPF_2, then there exist a joint, $L_{CPF-CPF}(CPF_1, CPF_2)$, at product family level.

Rule 7. If and only if there exist a bijective mapping for the mating relations between the part variants in an optional part family, OPF_1, and the part variants in another optional part family, OPF_2, and all the mating relations remain the same, then there exist a joint, $L_{OPF-OPF}(OPF_1, OPF_2)$, at product family level.

Rule 8. If and only if all the part variants in an optional part family, OPF_1, have the same mating relation with a personalized part, pp_1, then there exist a joint, $L_{OPF-PP}(OPF_1, pp_1)$, at product family level.

Rule 9. If a mating situation between the mating parts of a product family does not comply with the rules above, then the mating relations cannot be represented at the level of product family; they have to be represented at product variant level.

With the rules and definitions defined above, the DFC model is extended to accommodate the information of part interactions of the whole product family. The common mating relations at product family level are represented by the DFC models at product family level, while the varying mating relations at product variant level are represented by the DFC models at product variant level.

Figure 2-22 shows a DFC model for the gear pump family mentioned in subsection 2.1.1. The nodes in this DFC model are distinguished by different colors which represent the types of physical variety components in the PVDA of this gear pump family. According to the proposed rules and definitions, the liaisons between these variety components can be established and represented by the directed arcs and the undirected dash arcs in the figure. The directed arcs represent the mating relations between two variety components at product family level, which mean that the mating relations between all the mating parts from the two variety components remain the same with the mating relations represented by the directed arcs. The DOFs labeled on the directed arcs denote the number of DOFs constrained in the mating relations defined by the arcs. The small green circles on common part families and optional part families represent the feature clusters defined in the part families. Only the feature clusters of home-made part families are shown and the features of outsourcing parts are neglected for simplicity. Although all the mating relations between the mating parts of different product variants are the same, the associated 4 x 4 transformation matrices between the mating parts from the connected variety components could be different. The directed arcs also denote the joint type is mate for all mating parts from the two variety components, while the dash arcs represent contacts.

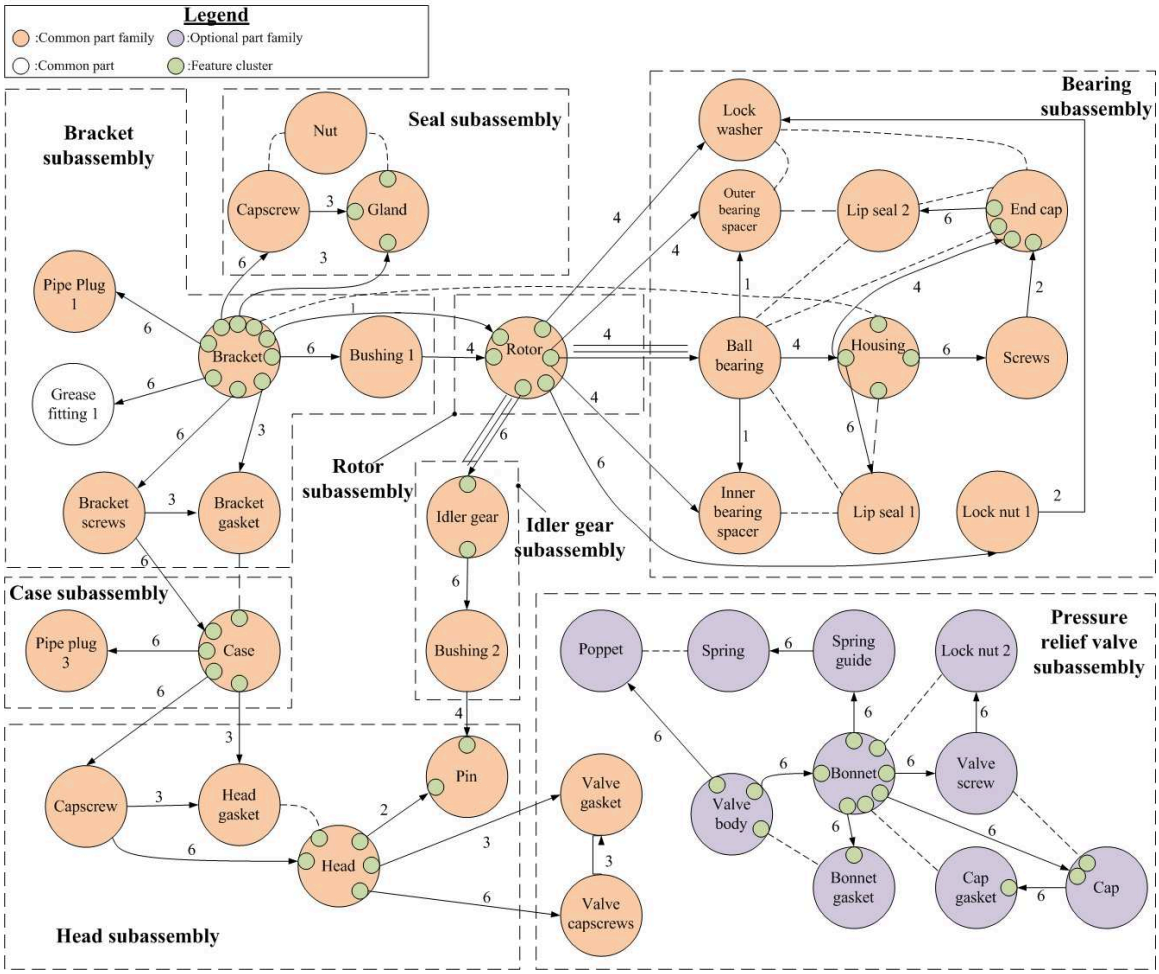


Figure 2-22: DFC model for the mating relations at product family level for the gear pump family

Figure 2-23 shows the mating relations represented by DFC model at product variant level for the gear pump family. Based on the rule 8, the mating relations between the part variants of a CPF and a PP and the mating relations between the part variants of a CPF and the part variants of an OPF should be represented at product variant level. For the part variants in a CPF, the part variants which are involved in the mating relations should be specified; for example, in the mating relations for pipe plug-1, bracket variants in the bracket family is specified to show which part variants have mating relations with the pipe plug-1. Because all the involved bracket variants have the same relations with the pipe plug-1, the node represent a set of bracket variants which consist of all the involved brackets; otherwise, the mating relations should be represented individually according to each involved bracket.

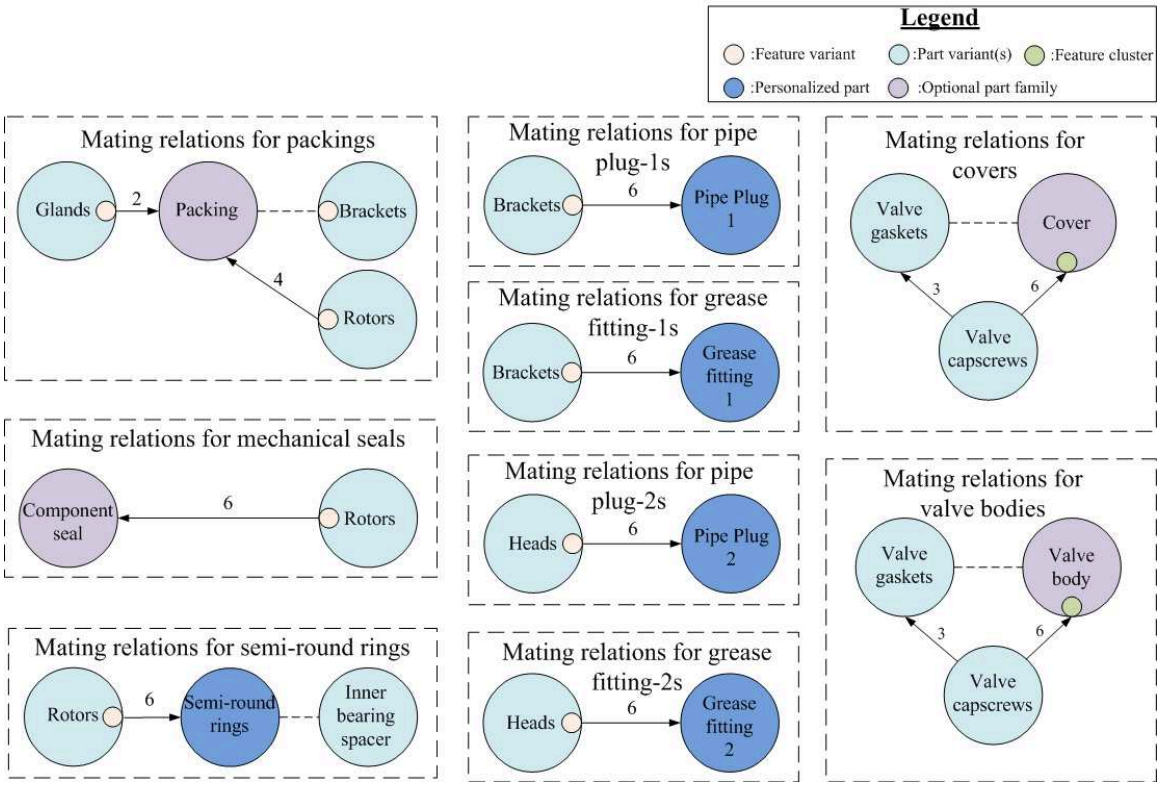


Figure 2-23: DFC model for the mating relations at product variant level for the gear pump family

The same representation way is applied for the mating relations between the part variants of a CPF and the part variants of an OPF; for example, in the mating relations for packings, because the involved glands variants have the same mating relation with the part variants in the packing family, the node for gland is a set which consists of all the involved gland variants, the node for packing is a set consisting of all the involved packing variants.

With the DFC models at product family level and at product variant level, the mating information in a product family is represented in a structural and concise way. The same mating relations between mating parts are represented only once. The mating information at product family level is shared by all the product variants, which can be used to generate the common part of assembly process plans for all the product variants; when a specific product variant is derived, the personalized mating relations for this specific product variant can be identified at product variant level, then the personalized part of assembly process plan can be added into the common part of assembly process plan to form the feasible assembly process plan for this specific product variant.

2.4 Conclusions

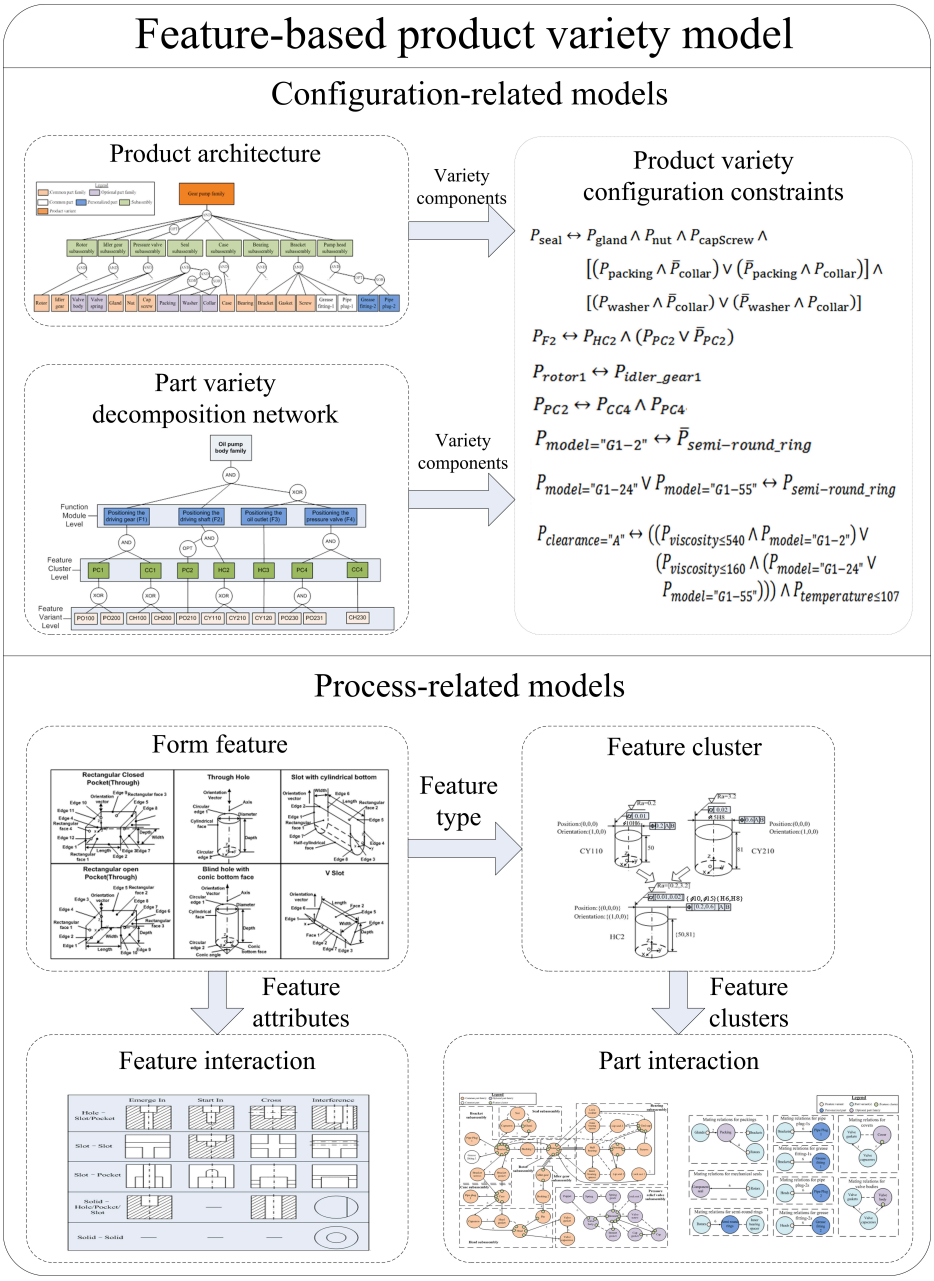


Figure 2-24: Relation framework of the proposed representation models

Computer-interpretable representation models of a product are necessary for computer-aided process planning. Traditional representation mechanisms represent each product/part individually. When it comes to a product family, those traditional modeling mechanisms incur large amounts of data redundancy, because there are similar attributes, similar components and similar structures among the product variants of the product family. Therefore,

the representation models for a product family which can handle the commonality and diversity inside the product family are indispensable for the generation of RPP. Feature-based modeling is an effective approach to represent the design specification of a product/part without introducing pure geometry. In this chapter, feature-based product variety models are proposed. Two principles are used when developing these representation models: 1)modularly decompose the product variety into correlative and configurable modules and use feature as the atomic representation; 2)maximally represent the commonality among all the variants in the family so as to reduce datum redundancy. The general relation framework of the proposed representation models are shown in figure 2-24.

Two aspects of product variety information are represented based on features: configuration related information and process-related information. For the representation of configuration-related information, product variety decomposition architecture is put forward, in which the structures of the variety components are represented at both product family's level and part family's level. A propositional-logic-based constraint representation mechanism is defined to describe the configuration constraints between the variety components in PVDA. At last, feature-based models are proposed to represent the process-related design specifications in a product family.

In the feature-based modes for a part family, a new concept - feature cluster is defined to capture the common specifications of a set of similar feature variants on different part variants in a part family; intervals and sets are used to aggregate the attribute values of the feature variants in a feature cluster.

One important aspect of process-related information is the interaction relations between features of a part variant and between mating parts of a product variant. For feature interactions, a knowledge-based representation approach is used to represent the interaction relations between features, and rules and situations are defined for the representation of the feature interactions at the level of feature clusters. For part interaction, rules and situations are defined to determine in which situation the mating relations can be represented at the level of product family; the mating relation represented at the level of product family are the common mating relations shared by all the relevant product variants; those mating relations which cannot be represented at product family level are represented at product variant's level; DFC models are used to represent the mating relations at product family's level and those at product variant's level.

In the following chapters, the configuration-related information represented in the feature-based product variety model is used for the configuration of new product variants; while the process-related information is processed for the generation of reconfigurable process plans for a product/part family.

Chapter 3

Reconfigurable process plan modeling

In contrast to conventional process plan which is designed to satisfy the manufacturing requirements of one single product or part, reconfigurable process plan satisfies the manufacturing requirements of all the product variants or part variants in a product family or a part family. In this section, the reconfigurable process plan for a product family whose variants are mechanical assemblies is defined as Reconfigurable Assembly Process Plan (RAPP), while the reconfigurable process plan for a part family whose variants are machining parts is defined as Reconfigurable Machining Process Plan (RMPP). The machining operation plans for feature clusters in a part variety decomposition network are defined as Reconfigurable Machining Operation Plan (RMOP). RMPP is built from a set of RMOPs with the sequence constraints on them. Both RAPP and RMPP are organized into modular process plan elements, each of which corresponds to a group of interactive variety components defined in the product variety decomposition architecture. AND/OR graph is adopted to represent each modularized assembly process plan in RAPP, and directed graph is used to represent each modular RMOP in RMPP.

3.1 AND/OR graph-based reconfigurable assembly process plan

Reconfigurable assembly process plan determines the assembly process plans(APP) of all the product variants in a product family. We give it the following definition:

Definition 22. Reconfigurable assembly process plan (RAPP) consists of a set of modular assembly process plan elements, each of which can satisfy all the assembly requirements of a group of interactive product variety components.

In the PVDA, physical variety components consist of four types of components: common parts, personalized parts, part variants in common part families and part variants in optional part families. As discussed in chapter 2, there are 10 situations of part interactions between these four types of components. According to these situations, the mating information between the mating parts is represented at two levels:

- Product family level. Chapter 2 defines 7 rules to determine the situations in which the mating information can be represented at product family level. The mating relation at this level is the common mating relation among the parts in all the involved product variants.

- Product variant level. Not all the mating situations between the interactive product variety components can be represented at the level of product family. Based on chapter 2, when the mating situations cannot comply with the rule 2 to rule 8, the mating relations in these situations have to be represented at product variant level. The mating information at this level refers to the mating relations among the parts in a specific product variant.

Because the mating relations at product family level and product variant level are incomplete mating relations in terms of a product variant, which means the DFC models for the mating relations at product family level and product variant level are not necessary to be connected (A graph is connected if and only if there is a path from any node to any other node in the graph). For example, in the DFC models for the gear pump family shown in figure 2-22 in chapter 2, the optional part families for the pressure relief valve subassembly are disconnected with the mating components of other subassemblies. The mating relations in the DFC models shown in figure 2-23 in chapter 2 are also disconnected with each other. Therefore, we divide the mating relations at the both levels into mating modules, each of which is a connected DFC model. The formal definition is given as follow.

Definition 23. A product family is a 8-tuple $< PrV, PA, CP, PP, CPF, OPF, R, F >$, where

- PrV is a set of all the product variants in the product family. No two elements of PrV correspond to the same product variant.

- PA is a set of all the parts of all the product variants in the product family. No two elements of PA correspond to the same part.
- CP is a set of all the common parts in the PVDA of the product family. No two elements of CP correspond to the same common part. $CP \subseteq PA$.
- PP is a set of all the personalized parts in the PVDA of the product family. No two elements of PP correspond to the same personalized part. $PP \subseteq PA$.
- CPF is a set of all the common part families in the PVDA of the product family. Each common part family CPF_i in CPF is a set of similar parts for every product variants. $CPF = \{CPF_1, CPF_2, .., CPF_i\}, CPF_i = \{p_1, p_2, ..., p_m\} \subset PA$. No two elements of CPF correspond to the same common part family.
- OPF is a set of all the optional part families is the PVDA of the product family. Each optional part family OPF_i is a set of similar parts for certain product variants. $OPF = \{OPF_1, OPF_2, ..., OPF_j\}, OPF_j = \{p_1, p_2, ..., p_n\} \subset PA$. No two elements of OPF correspond to the same optional part family.
- R is a set of relationships among the elements of $PrV \cup PA \cup CP \cup PP \cup CPF \cup OPF$. No two elements in R correspond to the same relationship. For example, $hasPart(prv_1, p_1)$ is a relationship that describes a product variant, $prv_1 \in PrV$, has a part, p_1; $hasPartVariant(CPF_1, p_1)$ is a relationship that describe a part, p_1, is a part variant in a part family, CPF_1; $L(p_1, p_2, prv_1)$ is a relationship that describes a part, p_1, has an assembly relationship with another part, p_2, in a product variant, prv_1; $L_{CP-CPF}(cp_1, CPF_1)$ is a relationship that describes a mating relation at product family level: a common part, cp_1, has an assembly relationship with a common part family, CPF_1.
- F is a set of attribute functions whose domains are subsets of $PrV \cup PA \cup CP \cup PP \cup CPF \cup OPF \cup R$. The functions in F associate entities or relationships to their characteristics such as the number of product variants, the parts in a specific part family, the type of a mating relationship (mate or contact) and the DOFs limited by a mating relationship.

Definition 24. MR is a 2-tuple $< MR_{PF}, MR_{PV} >$ which represents the mating information in a product family, where

- MR_{PF} is a set of mating information at product family level, each element $MR^i_{PF} \in MR_{PF}$ is a set of mating relations at product family level which form a connected DFC model. The components involved in a MR^i_{PF} which is called a mating module at product family level. No two elements in MR_{PF} correspond to the same mating information.

- MR_{PV} is a set of mating information at product variant level, each element $MR^i_{PV} \in MR_{PV}$ is a set of mating relations at product variant level which form a connected DFC model. The components involved in a MR^i_{PV} is called a mating module at product variant level. No two elements in MR_{PV} correspond to the same mating information.

In the example shown in figure 2-22, the MR_{PF} for the gear pump family has two mating modules. All common part families and a common part of this gear pump family form a connected DFC model, therefore their mating relations belong to one mating module, MR^1_{PF}. Another mating module at product family level, MR^2_{PF}, consists of the mating relations between all the optional part families because they form another connected DFC model. In the DFC model shown in figure 2-23, the MR_{PV} for the gear pump family has nine mating modules, each mating module is a connected DFC model for a specific optional part family or a personalized part family.

Based on the definition of reconfigurable assembly process plan, the assembly process plans in reconfigurable assembly process plan need to satisfy the assembly constraints both from the mating relations at product family level and the mating relations at product variant level. Therefore, reconfigurable assembly process plan is a set of assembly process plan elements, each assembly process plan element associates to a mating modules represented by a connected DFC model at product family level or at product variant level. We use AND/OR graph to represent the feasible assembly process plans for each assembly process plan element, thus, the representation of reconfigurable assembly process plan consists of a set of AND/OR graphs for all the mating modules in a product family.

Definition 25. The AND/OR graph of the feasible assembly process plans for a mating module whose interactive components are in a set M, is a 2-tuple $< S, T >$ where

- $S = \{\theta \in P(M) | \theta \neq \phi \wedge connected(\theta) \wedge stable(\theta)\}$ is the set of assembly states, $P(M)$ is the power set of M.

- T is the set of assembly tasks, in which the assembly task $< \theta_k, \theta_i, \theta_j >$ satisfies the following conditions:
  - $\theta_k = \theta_i \cup \theta_j \wedge geoFeasible(\theta_i, \theta_j) \wedge mecFeasible(\theta_i, \theta_j)$.
  - $\exists \theta_k (\theta_k = M)$.
  - $\forall < \theta_k, \theta_i, \theta_j > \in T$, if $|\theta_i| \neq 1$, then $\exists < \theta_x, \theta_y, \theta_z > (\theta_x = \theta_i)$; if $|\theta_j| \neq 1$, then $\exists < \theta_l, \theta_m, \theta_n > (\theta_l = \theta_j)$.

The elements in S are sets of interactive components, which represent the assembly states of assembly tasks. They are non-empty subset of the power set $P(M)$ and each of them is connected in the DFC model. A predicate - $connected(\theta)$ is used to determinate whether the non-empty subset is connected or not. A predicate - $stable(\theta)$ is used to determinate whether the non-empty subset is stable or not. An assembly state is stable if all the involved parts can maintain their relative position during the period of the whole assembly process.

Each assembly task in T is represented as $< \theta_k, \theta_i, \theta_j >$, in which θ_i and θ_j are two assembly states for two subassemblies, θ_k is the assembly state after joining the assembly state θ_i with the assembly state θ_j, thus $\theta_k = \theta_i \cup \theta_j$. A predicate - $geoFeasible(\theta_i, \theta_j)$ is used to determine whether the assembly task for θ_i and θ_j is geometrically feasible or not. An assembly task is said to be geometrically feasible if there is a collision-free path to move the interactive components of the two assembly states to the right assembly position from their original positions. A predicate - $mecFeasible(\theta_i, \theta_j)$ is used to determinate whether the assembly task for θ_i and θ_j is mechanically feasible or not. An assembly task is said to be mechanically feasible if there is no conflict during the establishment of the contacts between the interactive components.

With the definition of the AND/OR graph for a mating module, we define reconfigurable assembly process plan as a set of AND/OR graphs for each mating module in a product family.

Definition 26. A RAPP can be represented by a 2-tuple $< G_{PF}, G_{PV} >$, where

- G_{PF} is a set of AND/OR graphs, each of which associates to a mating module at product family level.
- G_{PV} is a set of AND/OR graphs, each of which associates to a mating module at product variant level.

For the gear pump family, the G_{PF} in the reconfigurable process plan $< G_{PF}, G_{PV} >$ consists of two AND/OR graphs, G^1_{PF}, G^2_{PF}. G^1_{PF} corresponds to the mating module, MR^1_{PF} and G^2_{PF} associates to the mating module, MR^2_{PF}.

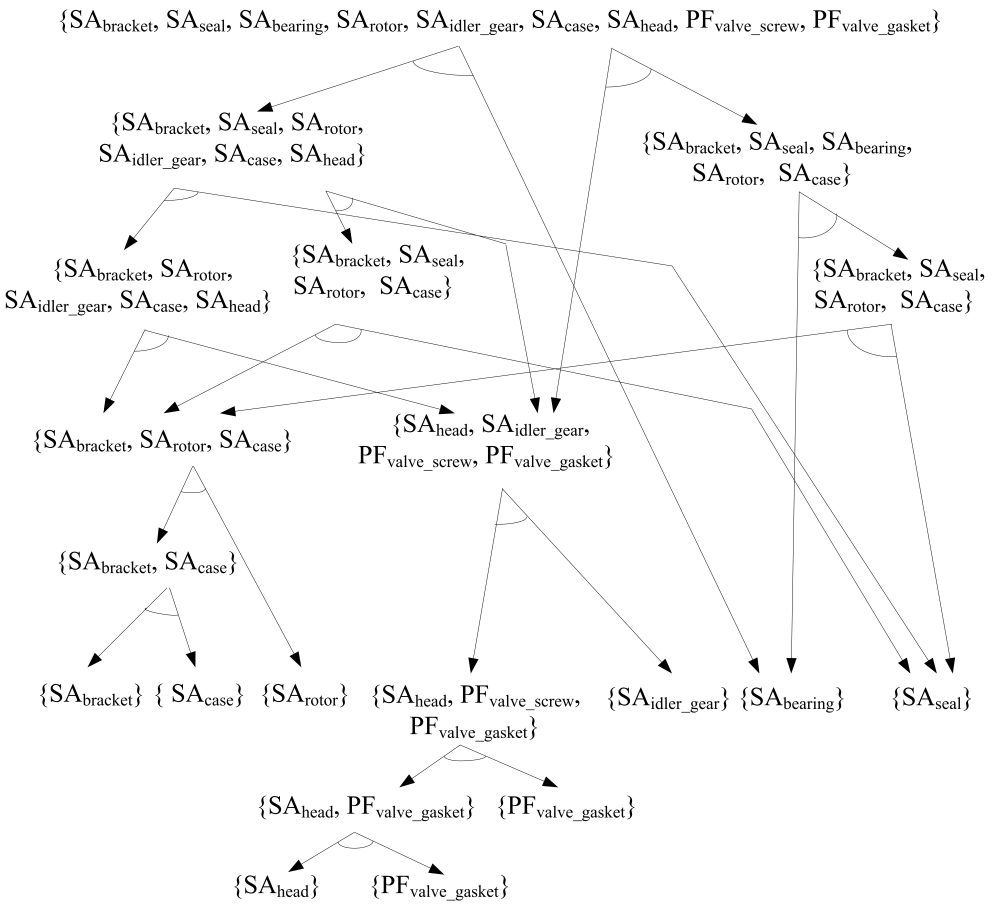


Figure 3-1: AND/OR graph, G^1_{PF}, for the mating module, MR^1_{PF}

Figure 3-1 shows the AND/OR graph, G^1_{PF}, for the mating module, MR^1_{PF}. For simplicity, only the interactive subassemblies which contain the interactive variety components are shown in the AND/OR graph. More detailed AND/OR graph can be constructed for the assembly of the interactive components in each subassembly. Figure 3-2 shows the AND/OR graph, G^2_{PF}, for the mating module, MR^2_{PF}. The nodes are the assembly states of a set of optional part families. The assembly tasks in the AND/OR graph are applicable for all the part variants of the part families. The assembly process plans in G^1_{PF} and G^2_{PF} are the feasible assembly process plans in terms of the mating parts in the mating modules, MR^1_{PF} and MR^2_{PF}, but they are not the final feasible assembly process plans for each gear pump variant in the gear pump family, because a part of mating information is represented at

product variant level.

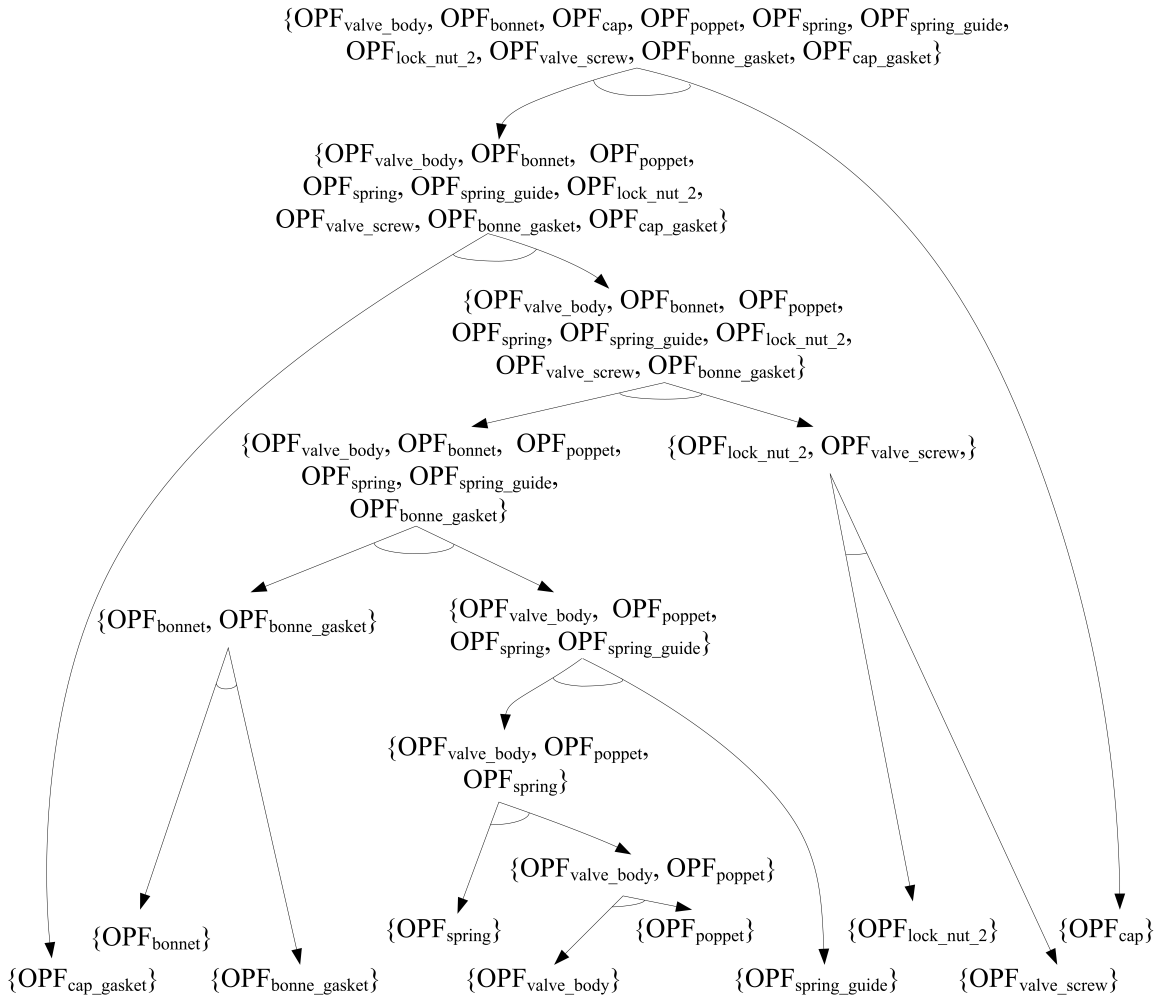


Figure 3-2: AND/OR graph, G^2_{PF}, for the mating module, MR^2_{PF}

The G_{PV} in the reconfigurable process plan for the gear pump family has 9 AND/OR graphs as its elements, each AND/OR graph associates to a mating module at product variant level. Figure 3-3 illustrates the AND/OR graphs in G_{PV} for the mating modules at product variant level. The aim of the AND/OR graphs in G_{PV} is to represent the assembly sequences for the interactive part variants which cannot be represented at product family level. The assembly tasks for the part variants from the mating modules at product family level are specified in G_{PF}, thus they are not necessary to be specified in the AND/OR graphs in G_{PV}. For example, the aim of G^1_{PV} is to specify the assembly tasks for all the part variants in an optional part family - *packing* family, which have mating relations with the part variants, *gland*, *rotor* and *bracket*, therefore, it is not necessary to specify the

assembly task for the assembly state $\{PV_{rotors}, PV_{bracket}\}$ in G^1_{PV}, because it can be found in the AND/OR graph, G^1_{PF}.

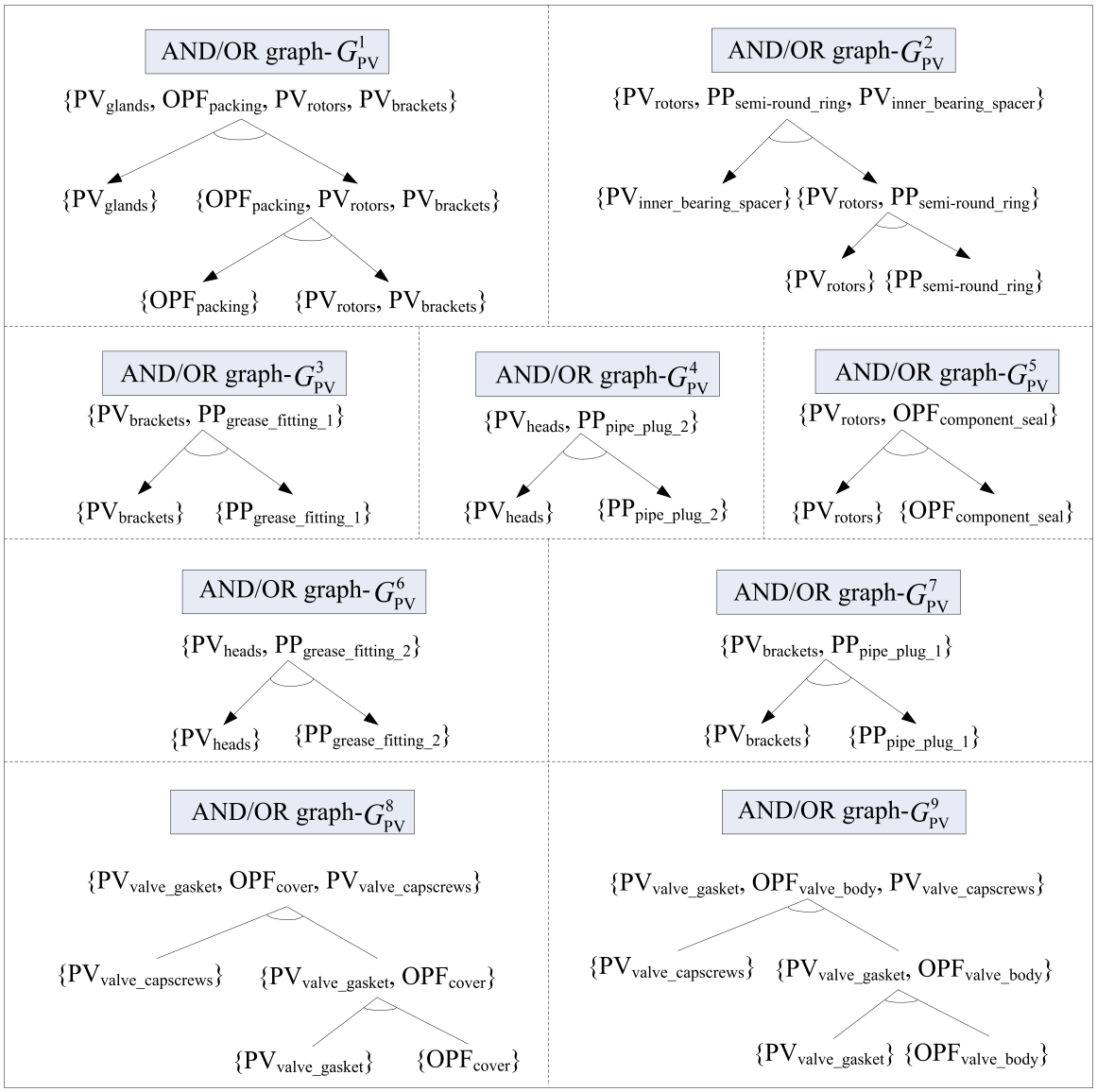


Figure 3-3: AND/OR graphs, G_{PV}, for the mating modules at product variant level

With a reconfigurable assembly process plan $< G_{PF}, G_{PV} >$, the feasible assembly process plans for a specific product variant are derived by combining the assembly process plans from G_{PF} with the assembly process plans from G_{PV}. The reconfigurable assembly process plan for a product family can be pre-generated without knowing which product variants would be put into production and what production resources would be available. The approaches for the generation of the reconfigurable assembly process plan for a product

family are discussed in the next chapter.

3.2 Directed graph-based reconfigurable machining process plan

Reconfigurable machining process plan (RMPP) is the RPP for a part family. The aim of RMPP is to provide feasible machining process plans for every part variant in a part family. As elaborated in chapter 2, a part family in a product family is either a common part family or an optional part family. The process-related information of part variants in a part family are represented by machining features and the commonalities of the machining features are represented by feature clusters. We give a mathematical definition for a part family as follows.

Definition 27. A part family is a 5-tuple $< FM, FC, FV, R, F >$ where

- FM is a set of function modules defined at the function module level in the part variety decomposition network. No two elements of FM correspond to the same function module.
- FC is a set of feature clusters defined at the feature cluster level in the part variety decomposition network. No two elements of FC correspond to the same feature cluster.
- FV is a set of feature variants defined at the feature variant level in the part variety decomposition network. No two elements of FV correspond to the same feature variant.
- R is a set of relationships among the elements of $FM \cup FC \cup FC$. No two elements of R correspond to the same relationship. For example, $and(\mathrm{FC}_1, \mathrm{FC}_2)$ is a configuration relationship on two feature clusters, FC_1 and FC_2; $hasAxis(\mathrm{fv}_1, \mathrm{g})$ is a relationship for geometric attribute between a feature variant fv_1 and a geometric element g; $emergeIn(\mathrm{FC}_1, \mathrm{FC}_2)$ is a relationship for feature interaction between two feature clusters, FC_1 and FC_2.
- F is a set of functions whose domains are the subsets of $FM \cup FC \cup FV \cup R$. The functions associate the entities or relationships to their characteristics, such as the type of a feature cluster, the attribute value of a feature variant, the involved feature variants in a feature interaction between feature variants.

In feature-based machining process planning, machining operations and their sequences for each machining feature are firstly determined, and then the machining process plans are generated by sequencing the machining operations of all the machining features on the part with the consideration of a set of sequencing constraints, including feature interaction, setup constraints, tool accessibility and good practice. Therefore, we introduce a new concept of reconfigurable machining operation plan (RMOP) for the feature clusters in a part family, and then we define RMPP as a set of RMOPs together with a set of feature precedence constraints.

Definition 28. Reconfigurable machining operation plan (RMOP) consists of a set of similar machining operation plans that satisfy all the machining requirements of all the feature variants in a feature cluster.

The similar machining operation plans are a set of machining operation sequences sharing a part of common machining operations and a part of the same precedence sequences. The machining operation plans in RMOP satisfy all the machining requirements of all the feature variants in a feature cluster. For any feature variants in the feature cluster, there exists at least one machining operation plan in which the machining operations together provide the integral machining capabilities that satisfy all the machining requirements of the feature variant. The machining requirements of a feature cluster correspond to the process-related design specifications of all the feature variants in this feature cluster, whose values are represented as intervals or sets.

The machining capabilities of a machining operation plan are the capabilities of the last operation in this machining operation plan, whose values can also be represented as intervals or sets. Therefore, a machining operation plan satisfies all the machining requirements of a feature cluster, if and only if the machining capabilities of this machining operation plan and the corresponding machining requirements of this feature cluster meet the following condition:

$$\forall r \in R, \exists c \in C(min(c) \leq min(r)) \tag{3.1}$$

where R is a set of machining requirements of a feature cluster, r is a machining requirement, C is a set of machining capabilities of a machining operation plan, c is a machining capability, $min(c)$ and $min(r)$ are the minimum values of c and r respectively.

Definition 29. A RMOP can be represented as a directed graph $G(V, E)$, where V is the

set of nodes and E is the set of ordered pairs of nodes called directed edges. The directed graph has the following properties:

- Each node in V is a machining operation selected to create the variants of a feature cluster;
- A directed edge written as (o_{p1}, o_{p2}) expresses that operation o_{p1} precedes operation o_{p2};
- There exists at least one starting node in $G(V, E)$. A node o_p is a starting node if there is a directed path from o_p towards other nodes of G, and no directed path to o_p.
- There exists at least one ending node in $G(V, E)$. A node o_p is a ending node if it does not precede any other nodes;
- A machining operation plan for a specific feature variant is a directed open path starting from a starting node and ending with the machining capability satisfying all the design specifications. In a machining operation plan, every node is visited at most once.

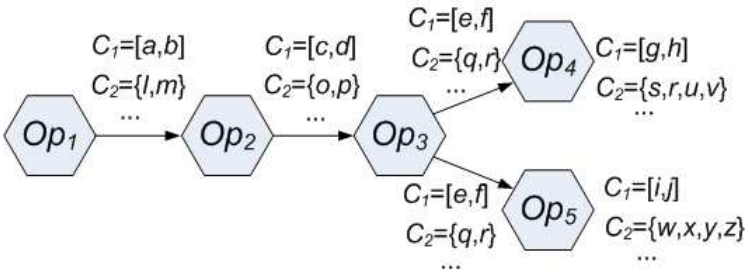


Figure 3-4: Reconfigurable machining operation plan

As shown in figure 3-4, the nodes correspond to the machining operations; the edges represent the machining priority between two nodes; the capabilities of the machining operation plan in the RMOP are labeled on the edges and the ending nodes, and the values are represented by using intervals or sets. The capabilities of a machining operations plan depend on the capabilities of the last machining operation in the sequence.

We use a hole cluster, *HC2*, of the oil pump body example shown in chapter 2 to illustrate the model of RMOP. The machining requirements of *HC2* are listed in table 3.1. Assuming that there exist four types of hole making operations which can be used to machine the

feature variants of *HC2*, they are *twist drilling*, *reaming*, *normal boring* and *precision boring*. These hole making operations can be found from a machining operation library by applying an operation selection technique. The machining capabilities of these hole making operations can also be extracted from the library, which are shown in table 3-2.

Table 3.1: The design specifications of a feature cluster: HC2

Position (R_p)	Orientation (R_o)	Surface roughness ($R_{sr}/\mu m$)	Geometrical specifications		Tolerance		
			Diameter (R_{dia}/mm)	Depth (R_{dept}/mm)	Dimensional tolerance(R_{dt})	True position (R_{tp}/mm)	Cylindricity (R_{cy}/mm)
$\{(0,0,0)\}$	$\{(1,0,0)\}$	$[0.2, 3.2]$	$\{\phi 10, \phi 15\}$	$\{50, 81\}$	$\{IT6, IT8\}$	$[0.2, 0.6]$	$[0.001, 0.02]$

Table 3.2: The machining capabilities of the hole making operations

	Twist drilling (O^{td})	Reaming (O^{r})	Normal boring (O^{nb})	Precision boring (O^{pb})
Tool diameter/mm	$[0.5, 88.9]$	$[1.6, 101.6]$	$[9.6, 304.8]$	$[9.6, 304.8]$
Depth/Dia limit	12	16	6	6
Surface roughness/μm	$[1.6, 6.3]$	$[0.2, 3.2]$	$[0.2, 6.3]$	$[0.2, 3.2]$
Dimensional tolerance/IT	$\{9...15\}$	$\{7...10\}$	$\{6...13\}$	$\{6...10\}$
True position/mm	$[0.05, 0.5]$	$[0.05, 0.3]$	$[0.03, 0.2]$	$[0.003, 0.2]$
Cylindricity/mm	$[0.1, 0.5]$	$[0.01, 0.1]$	$[0.01, 0.1]$	$[0.001, 0.01]$

By comparing the machining capabilities of the hole making operations in table 3.2 with the machining requirements of *HC2* in table 3.1, the RMOP for *HC2* is generated as shown in figure 3-5. The machining operation sequences in this RMOP are established by the predefined machining operation sequencing rules; the machining capabilities of a machining operation sequence are recalculated once a new machining operation is attached to the sequence; RMOP generation mechanism keeps adding more accurate machining operations into the sequence until all the machining requirements are satisfied by the capabilities of the operation sequence.

According to table 3.2, all of these four machining operations can satisfy the diameter and depth requirements of *HC2*; thus only the tolerance requirements are left to be satisfied by the machining operation sequences. The RMOP starts with twist drilling (the starting node) with which the machining operation sequence (o_{td}) only satisfies the location requirement of *HC2* ($min(c_{tp}^{o_{td}}) < min(r_{tp})$); then o_{td} is followed by *normal boring* and *reaming*, the

machining operation sequence ($o_{td} \Rightarrow o_r$) further improves the machining capabilities of the sequence and satisfies the surface roughness requirement of *HC2*, while the sequence ($o_{td} \Rightarrow o_{nb}$) further satisfies the dimensional tolerance requirement of *HC2* so does the sequence ($o_{td} \Rightarrow o_r \Rightarrow o_{nb}$). The machining operation sequences ending with o_{pb} ($o_{td} \Rightarrow o_r \Rightarrow o_{pb}, o_{td} \Rightarrow o_{nb} \Rightarrow o_{pb}$ and $o_{td} \Rightarrow o_r \Rightarrow o_n b \Rightarrow o_{pb}$) satisfy all the machining requirements of *HC2*. In a word, in the RMOP for *HC2*, there are three machining operation sequences satisfying all the machining requirements of the feature variants in *HC2*: $o_{td} \Rightarrow o_r \Rightarrow o_{pb}; o_{td} \Rightarrow o_{nb} \Rightarrow o_{pb}; o_{td} \Rightarrow o_r \Rightarrow o_{nb} \Rightarrow o_{pb}$, while the machining operation sequence satisfying the machining requirement of a specific feature variant in *HC2* can be derived from the directed graph of this RMOP. The algorithm used to automatically generate the RMOP of a feature cluster is proposed in chapter 4.

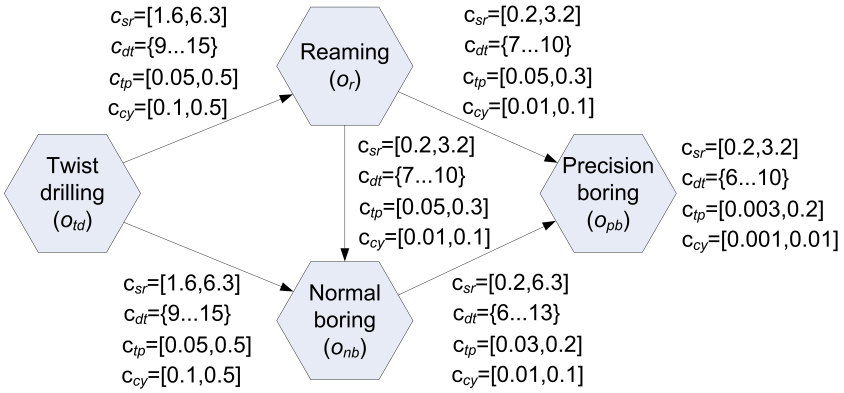


Figure 3-5: RMOP for a hole cluster: HC2

Definition 30. Reconfigurable machining process plan (RMPP) consists of the RMOPs corresponding to all the feature clusters of a part family and the precedence relations between the interactive part variety components. A RMPP is a 2-tuple $< \psi, \Omega >$,where

- ψ is a set of 2-tuples $< fc, G_{fc} >$, where
  - fc is a feature cluster of a part family, i.e. $fc \in FC$;
  - G_{fc} is a RMOP for the feature cluster fc represented as a directed graph.
- Ω is a set of precedence relations between the interactive part variety components(feature clusters or feature variants) of a part family, $\Omega = \{fv_1 \odot fv_2 | fv_1, fv_2 \in FV \ \wedge \ \odot \in \{softBefore, hardBefore, softImmeBefore, hardImmeBefore, equal\}\}$.

We define five types of precedence relations:

- A $softBefore$ B means the machining of feature variant A has to be started before the machining of feature variant B;
- A $hardBefore$ B means the machining of feature variant A has to be ended before the machining of feature B;
- A $softImmeBefore$ B means the machining of feature variant A has to be started immediately before the machining of feature variant B so that there is no other feature can be machined between A and B;
- A $hardImmeBefore$ B means the machining of feature variant A has to be ended immediately before the machining of feature variant so that there is no other feature can be machined between A and B;
- A *equal* B means feature variants A and B have to be machined simultaneously.

The precedence relations between the feature variants depend on the feature interactions, setup constraints, tool accessibility and good practice. As elaborated in chapter 2, feature interactions in a part family can be represented at two levels, feature cluster level and feature variant level. For the feature interactions at feature cluster level, as the feature interactions between all the interactive feature variants from the feature clusters remain the same, the precedence relations between these interactive feature variants could be the same if there was no other extra precedence constraints on the relationships. Therefore, the same precedence relations between the interactive feature variants can be represented at feature cluster level so as to reduce the redundancy. We have the following rule:

Rule 10. $\forall fc_1, fc_2((fc_1 \odot fc_2) \leftrightarrow \forall fv_1 \in fc_1, \forall fv_2 \in fc_2(fv_1 \odot fv_2))$

Table 3.3: Precedence relations between the feature variants of the two oil pump body variants

Oil pump body variant 1	Oil pump body variant 2
	$CY210$ $hardBefore$ $PO200$
	$CY210$ $hardBefore$ $PO210$
$CY110$ $hardBefore$ $PO100$	$CY210$ $hardBefore$ $PO231$
$CY110$ $hardBefore$ $CY120$	$PO200$ $hardBefore$ $CH200$
$PO100$ $hardBefore$ $CH100$	$PO230$ $hardBefore$ $CH230$
	$PO231$ $softImmeBefore$ $PO230$

In the oil pump body family example mentioned in chapter 2, the precedence relations between the feature variants of the two part variants are listed in table 3.3. As shown in table 3.3, $CY110$ should be machined before $PO100$ and $CY210$ should be machined before $PO200$. In the part variety decomposition network, $CY110$ and $CY210$ are the feature variants of a feature cluster: $HC2$; $PO100$ and $PO200$ are the feature variants of a feature cluster: $PC1$; therefore, these precedence relations can be represented at feature cluster level: $HC2\ hardBefore\ PC1$, which means that if a part variant have a feature variant, fv_1, from $HC2$ and a feature variant, fv_2, from PC1, then fv_1 should be machined before fv_2. In the same way, we have:

$$PC1\ hardBefore\ CC1,\ HC2\ hardBefore\ HC3,$$
$$HC2\ hardBefore\ PC4,\ PC4\ hardBefore\ CC4.$$

Based on Definition 30, the RMPP $< \psi, \Omega >$ for the oil pump body family mentioned in chapter 2 can be:

- $\psi = \{< PC1, G_{PC1} >, < CC1, G_{CC1} >, < PC2, G_{PC2} >, < HC2, G_{HC2} >,$
  $< HC3, G_{HC3} >, < PC4, G_{PC4} >, < CC4, G_{CC4} >\}$;
- $\Omega = \{HC2\ hardBefore\ PC1, PC1\ hardBefore\ CC1,$
  $HC2\ hardBefore\ HC3, HC2\ hardBefore\ PC4,$
  $PC4\ hardBefore\ CC4, PO231\ softImmeBefore\ PO230\}$.

$PC1$, $CC1$, $PC2$, $HC2$, $HC3$, $PC4$ and $CC4$ are the feature clusters in the part variety decomposition network for this oil pump body family. G_{PC1}, G_{CC1}, G_{PC2}, G_{HC2}, G_{HC3}, G_{PC4} and G_{CC4} are the RMOPs for each feature cluster of the oil pump body family respectively. Each RMOP is represented as a directed graph - $G(V, E)$. Figure 3-6 illustrates these RMOPs for the feature clusters of the oil pump body family under the consideration of a series of machining operations including: Milling, Drilling, Boring and Reaming.

As illustrated in the oil pump body example, RMOP for a feature cluster satisfies the machining requirements of all the feature variants in this feature cluster. By combining the RMOPs and the precedence relations, RMPP for a part family satisfies the manufacturing requirements of all the part variants in this part family. In addition, with the process-related design specifications of a part family, RMOP for the part family can be pre-generated before the configuration of a specific part variant; when a specific part variant is derived, the feasible

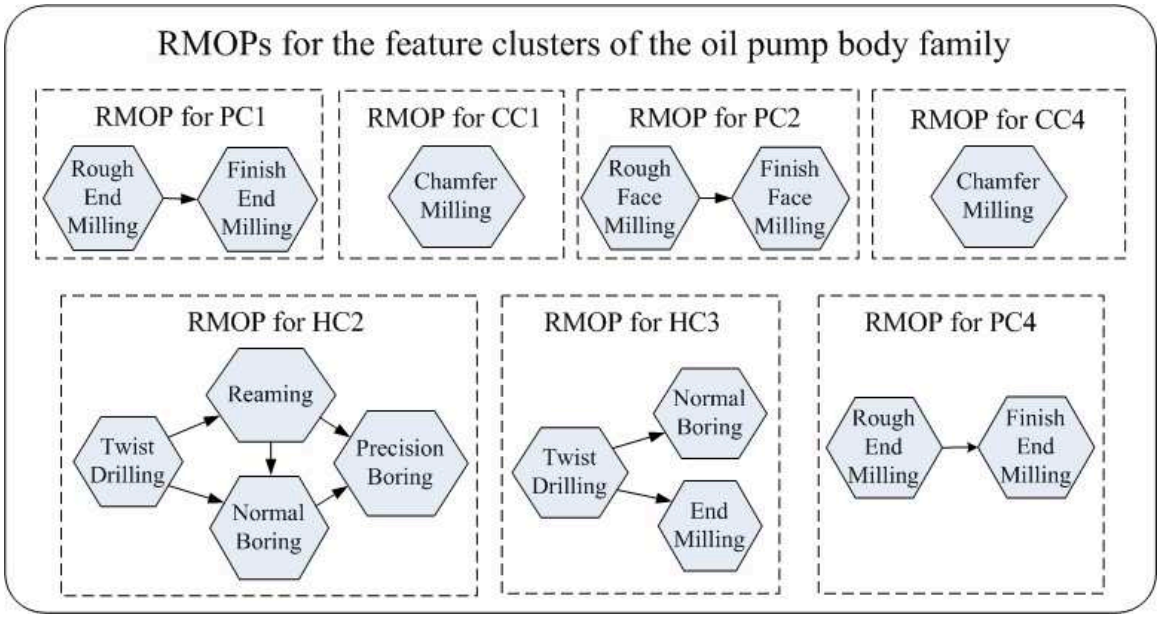


Figure 3-6: Illustration of the RMOPs for the feature clusters of the oil pump body family

machining process plans for this part variant can be generated from the RMPP instead of generating from scratch.

3.3 Conclusions

This chapter focuses on the models for the representation of reconfigurable process plan. Reconfigurable process plan is defined as a set of modular process plan elements, each of which satisfies all the manufacturing requirements of a group of interactive variety components. In response to the two levels of variety in a mechanical product system: product variety and part variety, two levels of reconfigurable process plan has been investigated: RAPP for a product family and RMPP for a part family. An AND/OR graph model is defined to represent the feasible assembly process plans for a mating module in a product family, then the RAPP for a product family consists of a set of AND/OR graphs, each of which associates to a mating module at either product family level or product variant level. For RMPP modeling, RMOP is defined for the feasible machining operation plans for the feature clusters of a part family, and a directed graph is used to represent a RMOP; RMPP for a part family consists of a set of RMOPs in combination with the precedence relations between the interactive part variety components (feature clusters/feature variants). The

gear pump family and the oil pump body family mentioned in chapter 2 are continued to be used as examples to illustrate the proposed models. This chapter only deals with the definitions and representation models in relation to reconfigurable process plan, while next chapter concerns the methods for the generation of reconfigurable process plan.

Chapter 4

Generation of reconfigurable process plan

This chapter deals with the approaches for generation of reconfigurable process plan. As shown in figure 4-1, it answers the question that how to use the process-related information represented by the product/part variety model proposed in chapter 2 to generate the reconfigurable process plan represented by the models proposed in chapter 3. Since the concept of reconfigurable process plan is defined at two levels: RAPP for a product family and RMPP for a part family, this chapter is devoted to propose the approaches and algorithms for the generation of RAPP as well as those for the generation of RMPP.

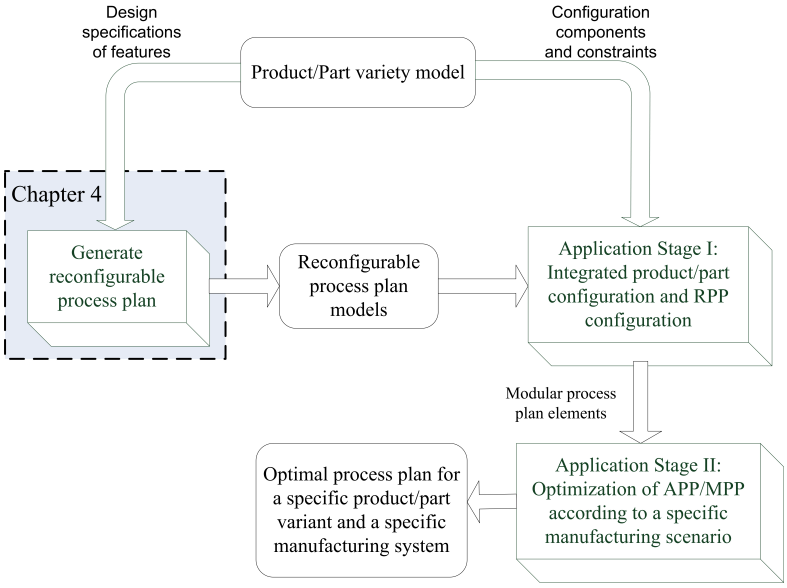


Figure 4-1: Framework of the thesis

4.1 Generation of reconfigurable assembly process plan

According to Definition 22, RAPP consists of a set of modular assembly process plan elements, each of which can satisfy all the assembly requirements of a group of interactive product variety components. Interactive product variety components could be a set of product variety components which have mating relationships with each other for one product variant or for several product variants. Four types of product variety components in a product family have been identified including common parts, common part families, optional part families and personalized parts.

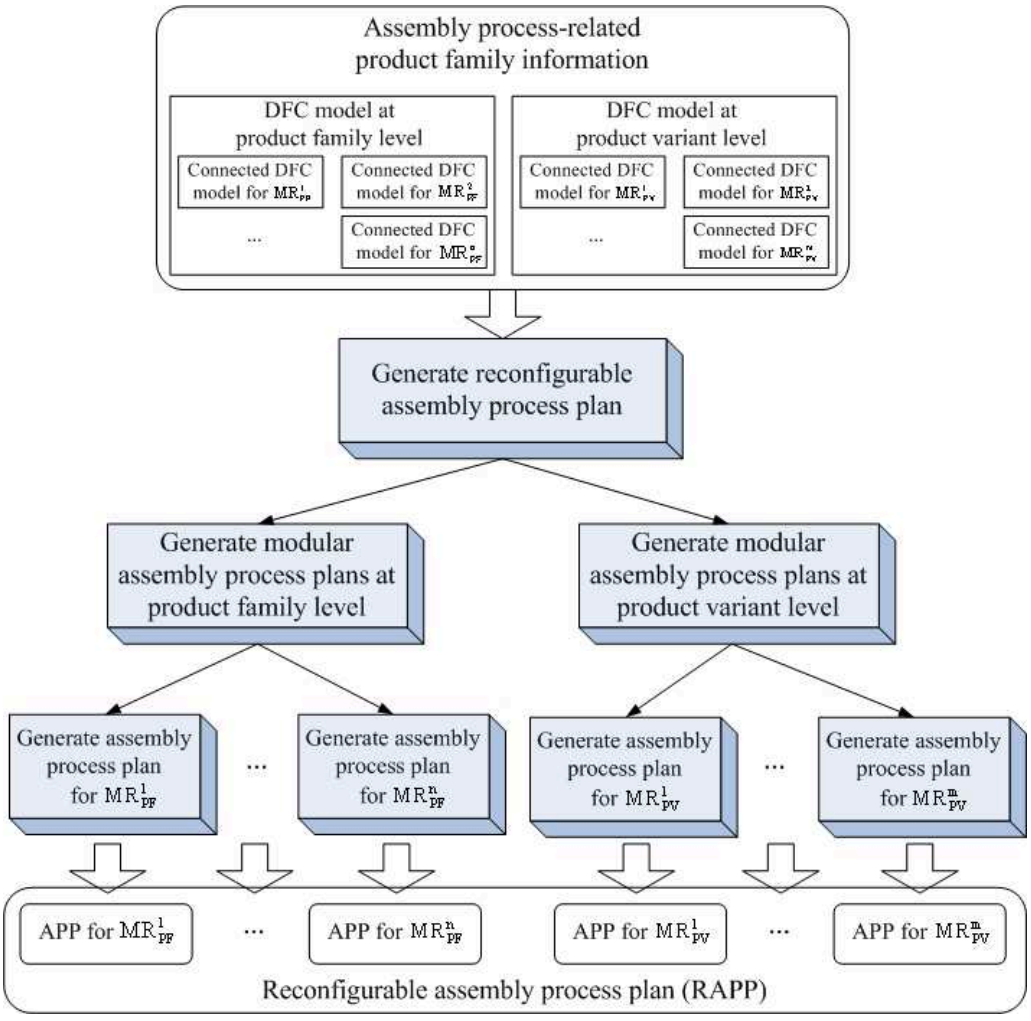


Figure 4-2: Decomposition procedure of the generation of reconfigurable assembly process plan

As RAPP is composed of modular assembly process plans, the generation of RAPP can be decomposed into the generation of every modular assembly process plan in the RAPP.

Figure 4-2 shows the decomposition procedure of the generation of RAPP.

RAPP is derived by analyzing the assembly-process related information of a product family. As mentioned in chapter 2, the assembly-process related information is represented by DFC models at both product family level and product variant level. Therefore, RAPP generation should consider the generation of modular assembly process plan at product family level as well as that at product variant level. In addition, as mentioned in chapter 3, each modular assembly process plan corresponds to a mating module whose mating information is represented by a connected DFC model, thus the generation of modular assembly process plan at product family level is decomposed into the generation of the assembly process plans for all the mating modules at product family level as well as at product variant level. The same representation models for the mating information at product family level and at product variant level allow that the same assembly process planning method can be applied at both levels.

The approaches for assembly process planning can be generally classified into two categories: 1) forward approaches; 2) disassembly approaches. In the forward approaches, the precedence constraints among the assembly operations for the liaisons inside an assembly are generated, and then the feasible assembly process sequences are derived by sequencing these assembly operations to satisfy all the precedence constraints and strategic constraints.

In the disassembly approaches, the generation of assembly sequences is transformed into the generating of disassembly sequences in which disassembly operations are the reverse of feasible assembly operations, the disassembly sequences are obtained through graph theory methods to recursively decompose an assembly into subassemblies.

Compared with the forward approaches, disassembly approaches have better computing efficiency because a subassembly directly implies the satisfaction of precedence relationship, whereas, in the forward approaches, the satisfaction of precedence relationship between a pair of mating parts may not be known immediately until an exhaustive search is completed (Lee, 1992).

Moreover, the algorithm for the disassembly approaches can guarantee to find solutions because an assembly should always be disassemble, while the algorithm for the forward approaches may encounter dead end if it is not guided by a proper backtrack mechanism. Therefore, we develop a disassembly approach to generate the modularized assembly process plans for the mating modules inside a product family.

4.1.1 A cut-set based disassembly approach for assembly process plan generation

The basic principle for the disassembly approach is to firstly decompose an assembly into a set of eligible subassemblies, and then to recursively apply the decomposition process to the subassemblies generated by the precious decomposition until no further decomposition can be applied. The inverse process of each decomposition from the subassemblies to their direct assembly forms one feasible assembly task in a feasible assembly process plan, and the inverse sequence among all the decompositions from the ended state (the parts of the assembly) to the initial state (the assembly) is the assembly process sequence of the feasible assembly tasks inside the feasible assembly process plan.

In our study, an assembly task refers to a series of assembly operations to transform two assembly states before its execution into one assembly state after its execution. A feasible assembly task needs to satisfy three aspects of assembly requirements:

- Geometric feasibility. An assembly task is said to be geometrically feasible if there is a collision-free path to move the interactive components of the two assembly states to the right assembly position from their original positions;

- Mechanical feasibility. An assembly task is said to be mechanically feasible if there is no conflict during the establishment of the contacts between the interactive components;

- Stability. Each sub-assembly produced from an assembly task must hold a stable assembly state. An assembly state is stable if all the involved parts can maintain their relative positions during the whole assembly process.

In order to guarantee the assembly tasks derived by the disassembly approach meet the assembly requirements (geometric feasibility, mechanical feasibility, and stability), the eligible subassemblies should satisfy the following conditions:

- Path existence condition. There exists a collision-free path to bring the subassemblies apart from mating;

- Local separable condition. All the liaisons between subassemblies can be separated from a common axis along a common direction;

- Stability condition. All the subassemblies resulted from a decomposition process should be stable which means all the liaisons inside every subassembly are maintained during the disassembly process.

For example, figure 4-3 shows a bearing assembly of a gear pump and its DFC model. In the DFC model, some parts, including *housing*, *ball bearing*, *end cap*, *inner bearing spacer*, *outer bearing spacer* and *screw*, have certain unconstrained DOFs, whereas *lip seal 1* and *lip seal 2* have no freedom DOFs. If the conditions described above are not considered, all these parts which have unconstrained DOFs could be an eligible subassembly for this bearing assembly. However, if the path existence condition is considered, *ball bearing* cannot be an eligible subassembly since there is no collision-free path to bring it apart from mating along one of its unconstrained DOFs; *ball bearing* also does not meet the local separable condition as the direction to separate the liaison with *inner bearing spacer* is opposite to the direction for separating the liaison with outer bearing spacer; if we consider the stability condition, *housing* is not an eligible subassembly because the liaisons among the rest of parts cannot be maintained after breaking all the liaisons of housing.

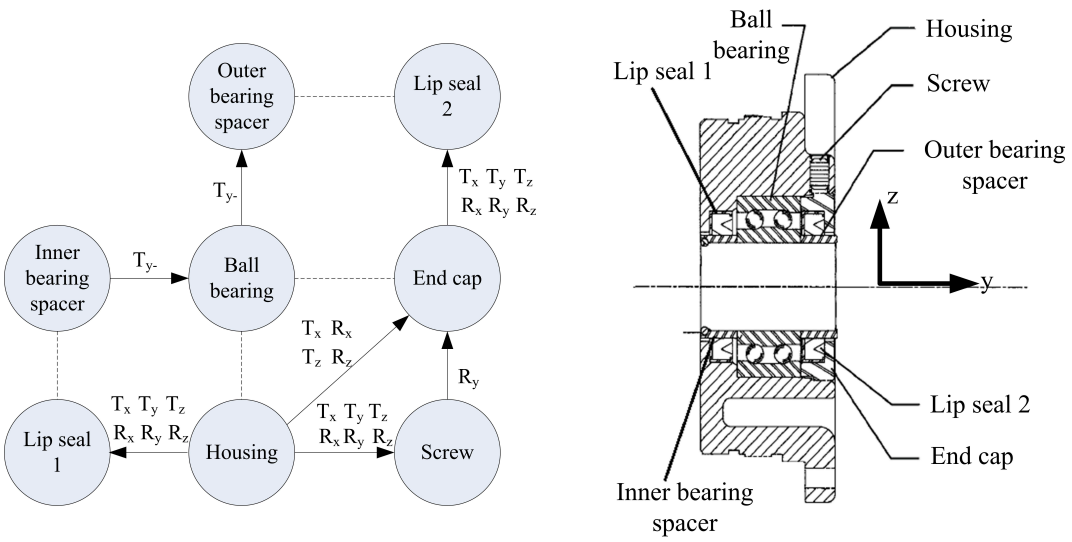


Figure 4-3: A bearing assembly of a gear pump and its DFC model

DFC model which represents the mating information of the interactive product variety components for a product family is analyzed to determine the eligible subassemblies. Therefore, the mathematical definitions for DFC model of a product family is given as follows.

Definition 31. A DFC model for a product family is a graph with attributed edges, $G_f =$

(V, E, A), where

- V is the set of vertices, each vertex corresponds to a product variety component of a product family;
- E is the set of edges, each edge corresponds to a mating relation between two product variety component; there are two types of edges in the graph, mate edge corresponds to the directed arc in DFC model and contact edge refers to the undirected dash in DFC model;
- A is the set of attribute functions on the edges, each attribute functions returns the specific attribute value(s) for a specific edge; for example, an attribute function - $type(e_1)$ returns the edge type of e_1, an attribute function - $dof(e_1)$ returns the DOFs constrained by the mating relation denoted by e_1.

Definition 32. A connected DFC model $G_c = (V_1, E_1, A_1)$ for a mating module in a product family is a component of $G_f = (V, E, A)$, where

- $G_c = (V_1, E_1, A_1)$ is a connected subgraph of $G_f = (V, E, A)$, such that all its vertices are connected and $V_1 \subset V$, $E_1 \subset E$, $A_1 \subset A$;
- E_1 consists of the edges of G_f that have one end vertex in G_c.

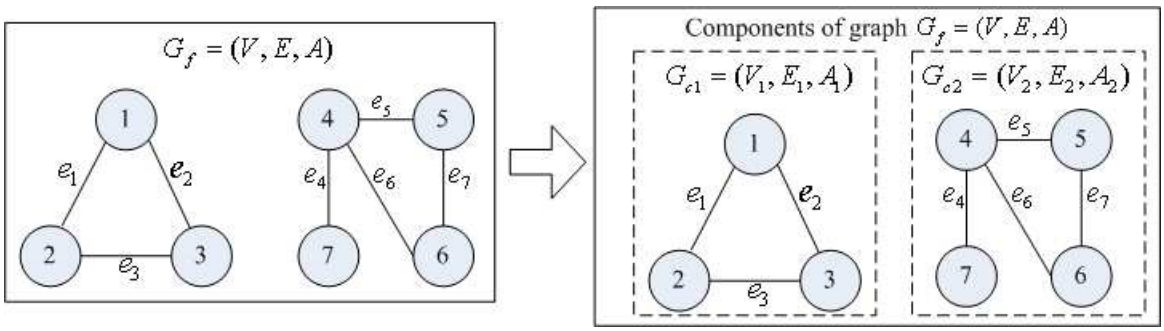


Figure 4-4: Illustration for Definition 32

Figure 4-4 shows an example for Definition 32. The DFC model $G_f = (V, E, A)$ consists of two components, $G_{c1} = (V_1, E_1, A_1)$ and $G_{c2} = (V_2, E_2, A_2)$, each of them satisfies the two conditions mentioned in Definition 32. With a connected DFC model - $G_c = (V_1, E_1, A_1)$ for a mating module, the eligible subassemblies can be identified through checking the above

conditions for the individual component defined by the cut-sets of $G_c = (V_1, E_1, A_1)$. According to graph theory, we have the following definitions:

Definition 33. A cut-set of a connected DFC model - $G_c = (V_1, E_1, A_1)$ is an edge set $F \subset E_1$ such that:

- $G - F$ (remove the edges of F in the graph G_c) is not connected, and
- $\forall H \subset F$, $G - H$ is connected.

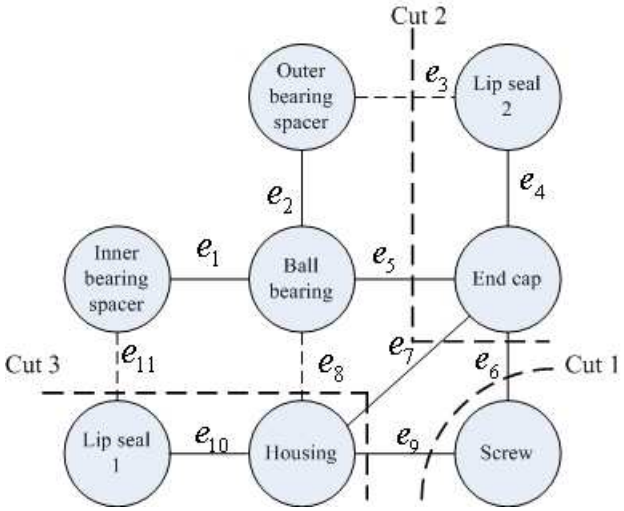


Figure 4-5: Three cuts of the DFC model of the bearing assembly

Definition 34. A cut of a connected DFC model - $G_c = (V_1, E_1, A_1)$ is a pair of subsets V_a and V_b such that $V_1 = V_a \cup V_b, V_a \cap V_b = \phi, V_a \neq \phi, V_b \neq \phi$.

In our work, it is assumed that each assembly task only realizes the mating liaisons between two subassemblies. Definition 33 guarantees that a cut-set of a connected DFC model - $G_c = (V_1, E_1, A_1)$ generates only two components of $G_c = (V_1, E_1, A_1)$ which is a cut of $G_c = (V_1, E_1, A_1)$. Figure 4-5 shows three cuts of the DFC model of the bearing assembly. For example, cut 1 is composed of {*screw*} and {*lip seal 2*, *end cap*, *outer bearing spacer*, *ball bearing*, *inner bearing spacer*, *lip seal 1*, *housing*}, and the cut set corresponding to cut 1 is $\{e_6, e_9\}$.

4.1.2 A pre-process procedure in the cut-set based disassembly approach

The problem of decomposing an assembly into the eligible subassemblies is equivalent to finding cut sets of the DFC model of the assembly. However, the number of all the possible

cut sets is an exponential function of the number of vertices, the maximum number could be $2^{|v|-1}-1$, where v is the number of vertices. In addition, a subassembly generated by a cut set may not be separable due to the accessibility and certain parts have the priority to be disassembled compared with other parts in the assembly, for instance, fasteners, like screw and nuts. In order to reduce the number of the vertices and to improve the correctness of the found solutions, a pre-process procedure is designed.

The preprocess procedure is composed of a series of test-and-mark processes and a reduced graph generation process. A flowchart for this pre-process procedure is shown in figure 4-6.

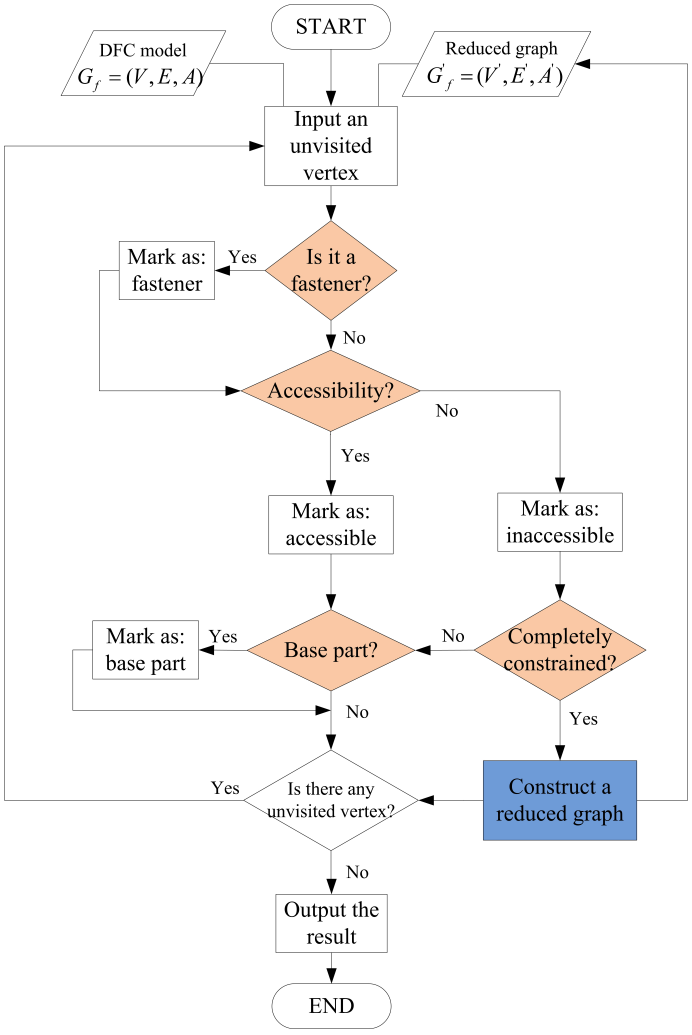


Figure 4-6: Flowchart for the preprocess procedure of the cutset-based approach

In the pre-process procedure, four decisions are necessary to be made to determine the key attributes of each part represented by each vertex in the DFC model. These key attributes

will then be checked during the generation of feasible cut sets to reduce the solution space.

The first decision is to determine whether the part denoted by the vertex is a fastener or not. In general, if a fastener is accessible, it should be disassembled firstly before all the other parts. Therefore, when searching for the eligible subassemblies for an assembly, a fastener and the subassembly consisting of the remaining parts are two potential eligible subassemblies.

The second decision is to determine whether the part denoted by the vertex can be accessible by a tool or hands if a disassembly operation is imposed on it. An inaccessible part is not an eligible subassembly because it makes the corresponding disassembly operation very difficult to execute. To determine the accessibility of a part requires not only the relative geometric information, but also the information (shape and motion) about the tools used in the disassembly operation. Therefore, it is very complex and difficult to calculate it automatically. In the current implementation, we adopt an interactive way to return the value of accessibility of a part.

The third decision is to determine whether the part (or component) denoted by the vertex in a DFC model is a base part (or based component) or not. In an assembly process, a base part is a part which gradually accommodates the other parts to form the final assembly. Thus, a base part cannot be disassembled until all the parts on it are disassembled. Normally, a base part is the heaviest, largest or most difficult to manipulate component (Delchambre, 1992). In some studies it is defined as the part with the most liaisons to other components (Ong and Wong, 1999; Danloy et al., 1999). We define the following base part conditions to test whether a part is a base part. The test is assisted by the DFC model.

In a DFC model, if there exists a vertex,v, that satisfy the following conditions, then it is a base part:

- v is not a fastener;
- The edges connecting to v are either mating-type edges with their tails connecting to it or contact-type edges;
- By removing this vertex, the remaining components in this DFC model become unstable.

For the bearing assembly illustrated in figure 4-3, the part,*housing*, is a base part, because it satisfies with the three base part conditions. Firstly, it is not a fastener; secondly, it is

connected by three mating-type edges with tails starting from it and one contact-type edge, in other words, *housing* serves as a location host for other four parts; in addition, if all the edges of *housing* are disconnected, the remaining component would become unstable.

The fourth decision is to determine which vertex can be merged into a compound vertex together with the vertex connecting with it. This decision can reduce the number of vertices of a DFC model and lead to a reduced graph. In an assembly, certain parts cannot be separated before other parts are separated. Thus, these parts can be merged into one compound vertex in a reduced graph so as to reduce the solution space for searching the feasible cut sets of the original DFC model. A vertex has to satisfy two conditions to be merge-able:

- it is inaccessible so that it cannot be separated before the parts covering on it are disassembled;
- all its DOFs are constrained by one part/component so that it is stable on the part.

The merged vertices form a component which is represented by a compound vertex in a reduced graph. Two vertices are merged once at a time, one is the vertex currently being checked and satisfying the merge-able conditions, another one is the vertex connecting to it and constraining all its DOFs.

Figure 4-7 illustrates how to merge two vertices into one compound vertex and then form a reduced graph. To merge two vertices, it is simply remove the two vertices and the edges between them, and then replace them with a compound vertex in the original graph. To construct a reduced graph, it is to retain the edges whose one end is connected to one of the merged vertices and the other end is connected to any other vertices except the merged vertices, and then to connect them to the compound vertex. The pre-process procedure will merge all the merge-able vertices, thus the reduced graph will be updated several times until there is no merge-able vertices in the graph. The merged vertices including the original edges between them will be revisited when the decomposition of the reduced graph is finished. In figure 4-6, there are two merge-able vertices in the DFC model after testing them with the two conditions, therefore in the final reduced graph, there are two compound vertices. Compared with the original DFC model, the number of the vertices is decreased by two in the reduced graph. The final reduced graph will be used for searching the feasible cut sets of the corresponding assembly.

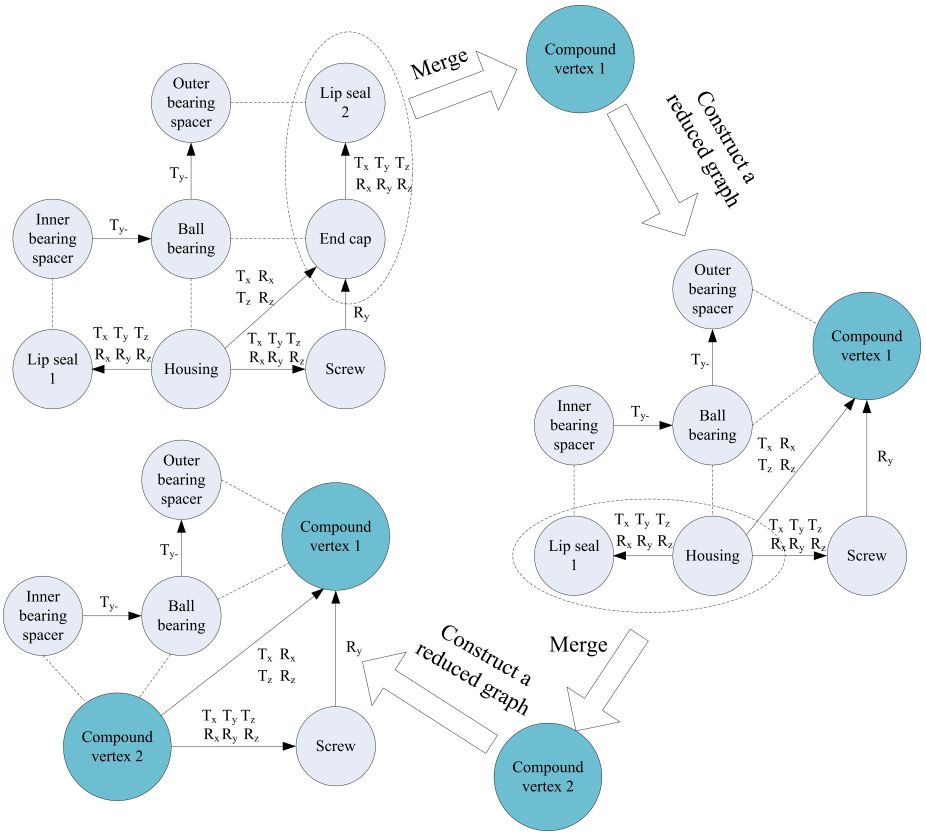


Figure 4-7: Merging two vertices and constructing a reduced graph

4.1.3 A general algorithm for the cut-set based disassembly approach

In this section, a general algorithm is proposed to realize the cut-set based disassembly approach. The input of this algorithm is a connected DFC model for a mating module of a product family; the output is the feasible assembly process plan represented by an AND/OR graph for the mating module.

Figure 4-8 shows the flowchart of the algorithm. It starts from the pre-process procedure mentioned in the previous subsection, which aims to reduce the solution space of cut-set searching process. The pre-process procedure derives a reduced graph which is used as an input for the following searching process.

Following the pre-process procedure, accessible fasteners are checked firstly. These fasteners directly lead to a series of cuts for the mating module. Each cut consists of a fastener, v_i, and a stable subassembly, $V^{'} - v_i$, constituted by the rest of parts in the reduced graph, $G_f^{'} = (V^{'}, E^{'}, A^{'})$. Once a cut is found, the algorithm will add the cut into the AND/OR graph. It follows the following form to represent a cut in a AND/OR graph:

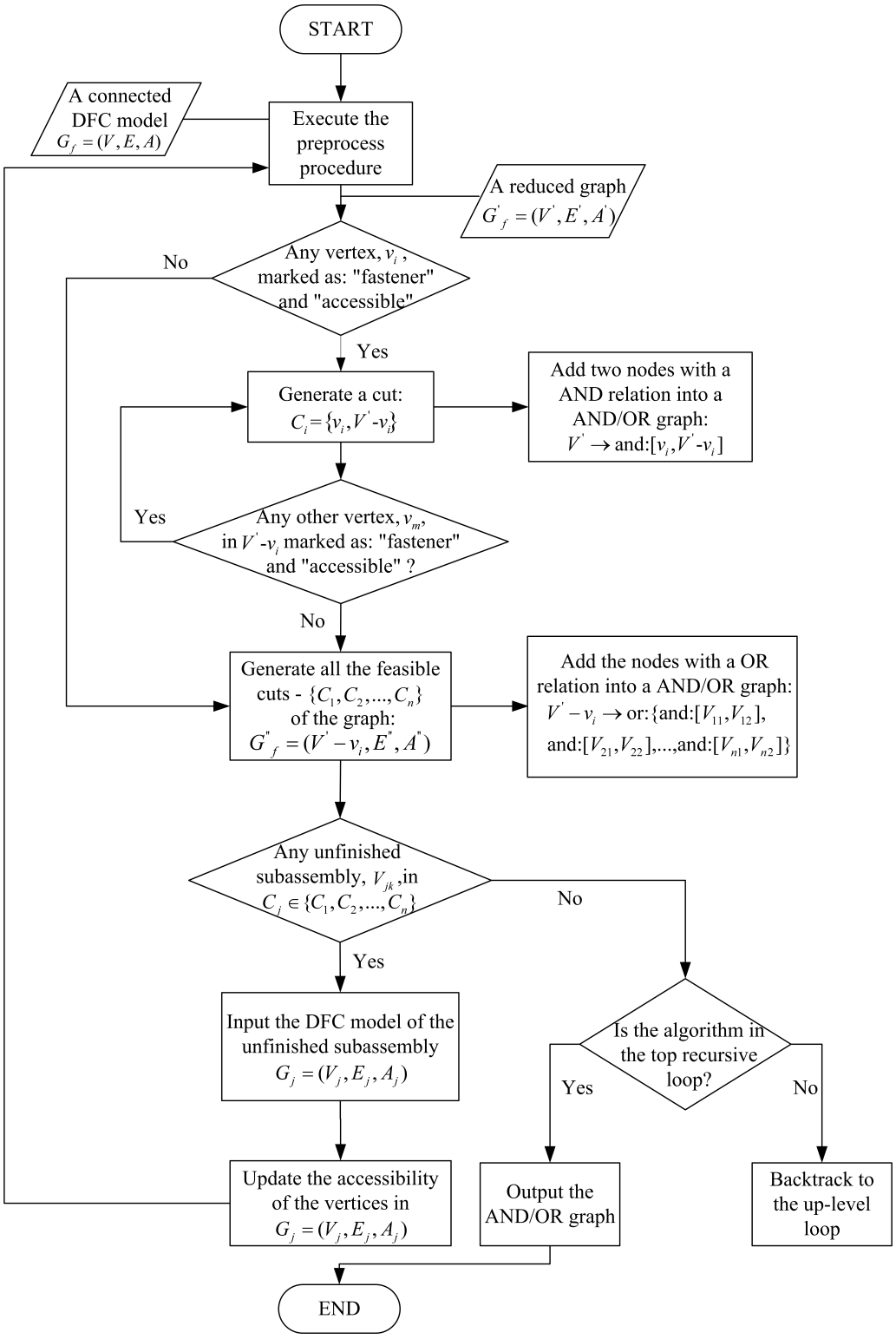


Figure 4-8: Flowchart of the general algorithm for the cut-set based disassembly approach

$$V^{'} \rightarrow and : [v_i, V^{'} - v_i],$$

where $V^{'}$ is a set of vertices before the cut; v_i and $V^{'} - v_i$ are the two components of the cut, they are also the two subsets of $V^{'}$; "$\rightarrow$" represents the hyperarc connecting the parent node: $V^{'}$ to the child nodes v_i and $V^{'} - v_i$ in the AND/OR graph; "$and : [v_i, V^{'} - v_i]$" represents that v_i and $V^{'} - v_i$ has **"and"** relation in the AND/OR graph, which means the assembly $V^{'}$ can be disassembled into two eligible subassemblies that v_i and $V^{'} - v_i$.

After recursively processing all the accessible fasteners, a new reduced graph - $G_f^{''} = (V^{''}, E^{''}, A^{''})$ is derived, and then it is used by the following process to search the feasible cuts. The process - Generate all the feasible cuts invokes a cut-set based algorithm to searching for all the feasible cuts which satisfy the three conditions for the eligible subassemblies. The cut-set based algorithm returns all the feasible cuts of the reduced graph - $G_f^{''} = (V^{''}, E^{''}, A^{''})$, and then these cuts are added into the AND/OR graph with the following form:

$$V^{''} \rightarrow or : \{and : [V_{11}, V_{12}], and : [V_{21}, V_{22}], ..., and : [V_{n1}, V_{n2}]\},$$

where $V^{''}$ is a set of vertices in the reduced graph - $G_f^{''} = (V^{''}, E^{''}, A^{''})$ passed from the previous process; assume that the feasible cuts found by the process are $C_1, C_2, ..., C_n$, then V_{11} and V_{12} are two subsets defined by C_1 such that $V_{11} \cup V_{12} = V^{''}$ and $V_{11} \cap V12 = \phi$, V_{21} and V_{22} are the similar two subsets defined by C_2, and V_{n1} and V_{n2} are another pair of subsets defined by C_n. The logic relation between these pairs of subsets defined by different feasible cuts is **"or"** which means there exist different ways to disassembly the assembly - $V^{''}$.

Figure 4-9 illustrates the process to construct a AND/OR graph in the general algorithm. After adding the found cuts into a AND/OR graph, the algorithm checks these cuts to find that if there is any unfinished subassembly in it. A subassembly which consists of more than one parts is an unfinished subassembly. If it is true, the algorithm continues with this unfinished subassembly and updates the accessibility of each part of the subassembly, then repeats the previous processes starting from the preprocess procedure; otherwise, the algorithm continues with a judgment: is the algorithm in the top recursive loop? If it is true, then it outputs the final AND/OR graph; otherwise, it backtracks to the up-level loop to find the next unfinished subassembly and repeats the following processes. In order

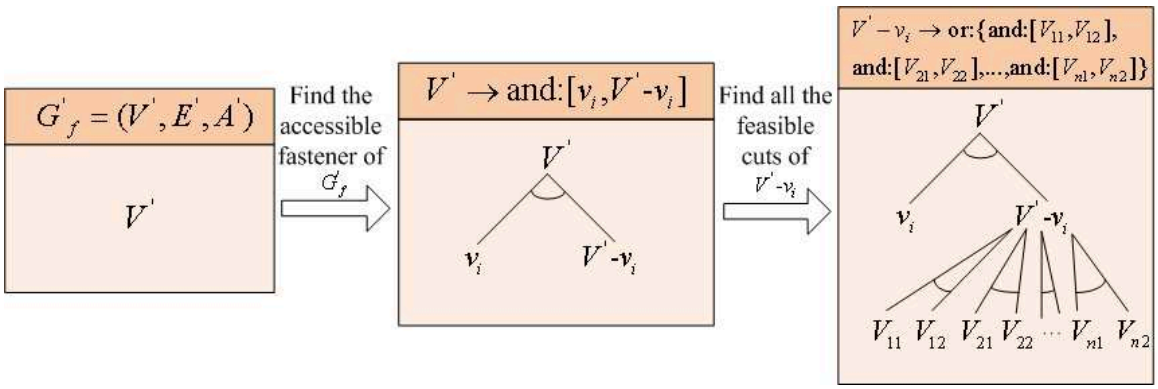


Figure 4-9: Illustration for the construction of a AND/OR graph in the general algorithm

to traverse all the unfinished subassemblies at every node level of a AND/OR graph, a backtrack mechanism is deployed. The general algorithm ends until there are only two parts in all the cuts and returns the final AND/OR graph for the mating module.

4.1.4 A guided Karger's algorithm for the generation of all the feasible cuts

After removing all the feasible fasteners, the general algorithm searches for all the feasible cuts for a reduced graph, $G''_f = (V'', E'', A'')$, the feasible cuts are composed of the eligible subassemblies of the assembly V''. The proposed algorithm for the generation of all the feasible cuts is inspired by Karger's contraction algorithm(Karger, 1993).

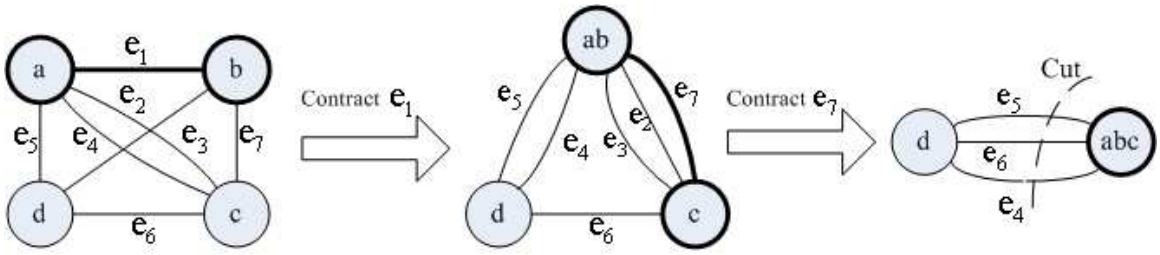


Figure 4-10: Contraction of edges in Karger's algorithm

Karger's algorithm is a randomized algorithm based on the concept of contraction of an edge (a, b) in an undirected graph $G = (V, E)$. To contract an edge (a, b), it follows the processes:

- Delete all edges between a and b;

- Merge a and b into a new super-vertex ab;
- Replace all the edges incident to a or b with edges incident to the super-vertex ab.

The algorithm iteratively contracts randomly chosen edges and continually reduces the number of vertices of the graph until only two vertices left. The edges crossing these two vertices form the cut-set and the two vertices represent a cut. Figure 4-10 illustrates the process of Karger's algorithm.

The proposed algorithm adopts the principle of edge contraction in Karger's algorithm but the edges are chosen by following predefined rules instead of randomly chosen edges. The aim is to contract the edges that cannot be disconnected because of inseparability. Therefore, we define the following rules to guide the algorithm to choose the edges to contract.

Rule 11. If there exists an inseparable edge in the reduced graph, then this edge should be contracted.

Explanation: The parts connected by an inseparable liaison must belong to an eligible subassembly; therefore the corresponding edge in the reduced graph should be contracted during the cut-set searching process. To identify an inseparable edge, it needs to check both the vertices connected by that edge. If both of them have no separate direction as equal to one of their DOFs, then the edge is an inseparable edge.

Rule 12. If there exists multiple edges between two vertices, then these multiple edges should be merged into one super-edge.

Explanation: In the case of multiple edges between two vertices, no matter which edge among the multiple edges is picked for the contraction, the result would be the same; therefore the multiple edges should be merged into one super-edge so as to avoid generating one cut several times. Figure 4-11 illustrates the process for merging multiple edges in the proposed algorithm.

Rule 13. If one of the multiple edges between two vertices is inseparable, then the merged super-edge is inseparable.

Explanation: In this case, one of the multiple edges is inseparable which means that one of the liaisons between two parts cannot be disconnected; therefore these two parts cannot be disassembled and the super-edge should be inseparable.

Figure 4-12 shows the flowchart of the proposed algorithm for the generation of all the feasible cuts. The three rules mentioned before have been applied into the algorithm to

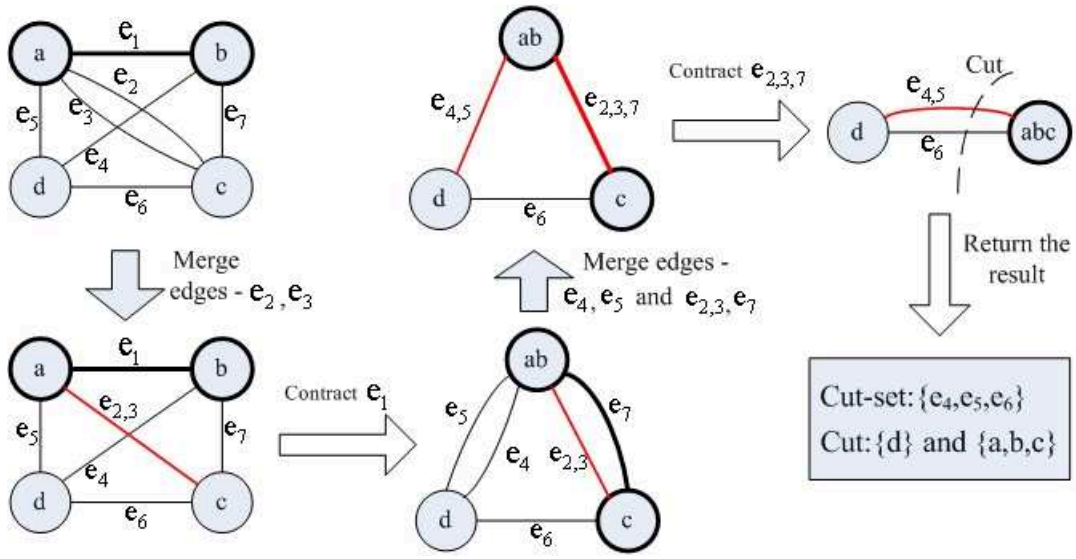


Figure 4-11: Illustration of merging multiple edges in the proposed algorithm

guide it to contract edges. In addition, a backtrack mechanism has been introduced into the algorithm to guide it to explore the whole solution space. In order to avoid repeatedly generating the same reduced graph, a new generated graph is compared with the previous generated graphs and only a new graph can be used for further exploration. A database is used for storing the different reduced graph generated during the execution of the algorithm. When a graph containing only two vertices has been found, the algorithm then applies a test to judge whether the subassemblies determined by the cut of this graph are eligible or not.

The algorithm starts with an input of a reduced graph, $G''_f = (V'', E'', A'')$. Then it checks the separability of all the edges in G''_f and marks the results for each edge. After that, it contracts all the edges marked as inseparable, and then generates a new reduced graph, $G''_{f1} = (V''_1, E''_1, A''_1)$, which has no inseparable edges in it.

In the next step, it merges all the multiple edges in G''_{f1} so as to reduce the solution space. After this step, it contracts an unvisited edge in the graph, and then compares the generated reduced graph with the previous generated graphs. If one of the previous generated graphs has the same vertices with the generated reduced graph, this generated reduced graph will be discarded, and then the algorithm will check whether there is still an unvisited edge in the reduced graph, if yes, the algorithm continues to contract this unvisited edge and repeats the comparison process; otherwise, the algorithm backtracks to the previous reduced graph right before current reduced graph and repeats the process to check whether there is still an

unvisited edge in the previous reduced graph.

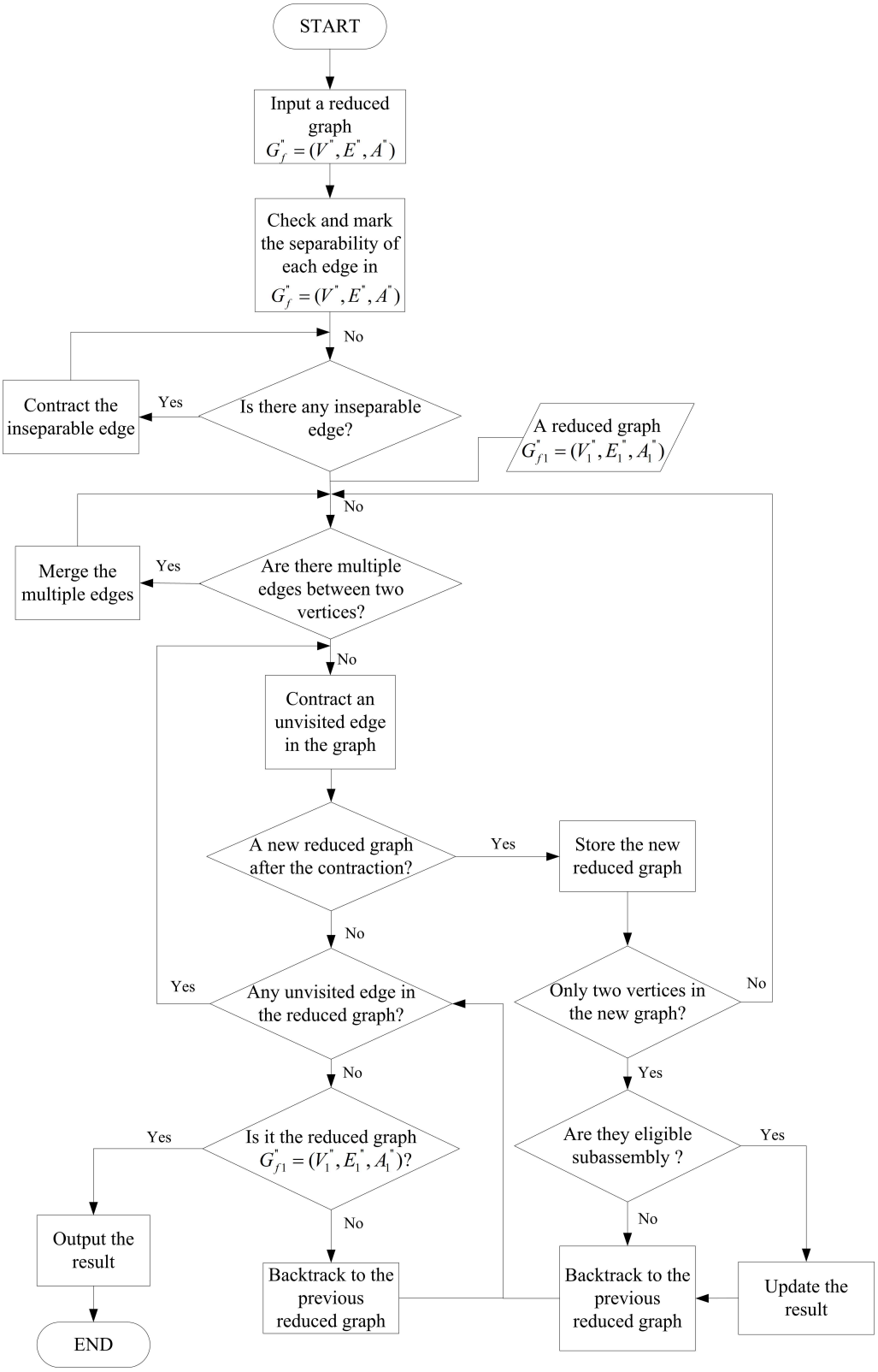


Figure 4-12: Flowchart of the proposed algorithm for the generation of all the feasible cuts

If the process for edge contraction generates a new reduced graph that are different with all the previous generated graphs, then the new reduced graph is stored into a database for the future comparison. If the new reduced graph only consists of two vertices, then a cut of the original graph G_f'' has been found and the algorithm will check the feasibility of this cut through testing the stability condition and path existence condition of the subassemblies determined by the cut. If they are eligible subassembly, then the cut is a feasible cut which will be updated into result, then the algorithm continues to find other feasible cut by backtracking to another unvisited edge in the reduced graph. If the new reduced graph consists of more than two vertices, then it needs to be further contracted; therefore, the algorithm will continue to merge the multiple edges in the graph and then to contract an unvisited edge again.

When the algorithm backtracks to the reduced graph G_{f1}'' and all its edges are visited, then the algorithm outputs the result and terminates itself.

Figure 4-13 shows an example to apply the proposed algorithm to search the feasible cuts of the bearing assembly mentioned in the previous section. The proposed algorithm explores the whole solution space and finds three feasible cuts for the bearing assembly. It starts with a reduced graph of the bearing assembly. After contracting the inseparable edges and merging the multiple edges, the algorithm continues to iteratively contract the unvisited edges in corresponding reduced graph. Only a new reduced graph will be further explored by the algorithm, otherwise, the algorithm backtracks to find another unvisited edge to contract. When there is no unvisited edge or a cut is found, the algorithm also backtracks to the previous reduced graph. In this example, the algorithm backtracks 8 times to explore the whole solution space. At last, once a cut is found, its feasibility is checked through testing the stability condition and path existence condition of the subassemblies determined by the cut. For example, in terms of the cut 1, the two subassemblies determined by it are {*lip seal 2* (ls2), *housing* (h), *ball bearing* (bb), *inner bearing spacer* (ibs)} and {*lip seal 1* (ls1), *end cap* (ec), *outer bearing spacer* (obs)}. These two subassemblies are both stable subassemblies by analyzing their DOF constraints represented in the DFC model; and a collision-free path can be found to bring these two subassemblies apart from mating; they also satisfy the local separable condition because the edges in the cut-set of cut 1, *e2* and *e6*, are separable edges. Therefore, the two subassemblies are eligible subassemblies and cut 1 is a feasible cut.

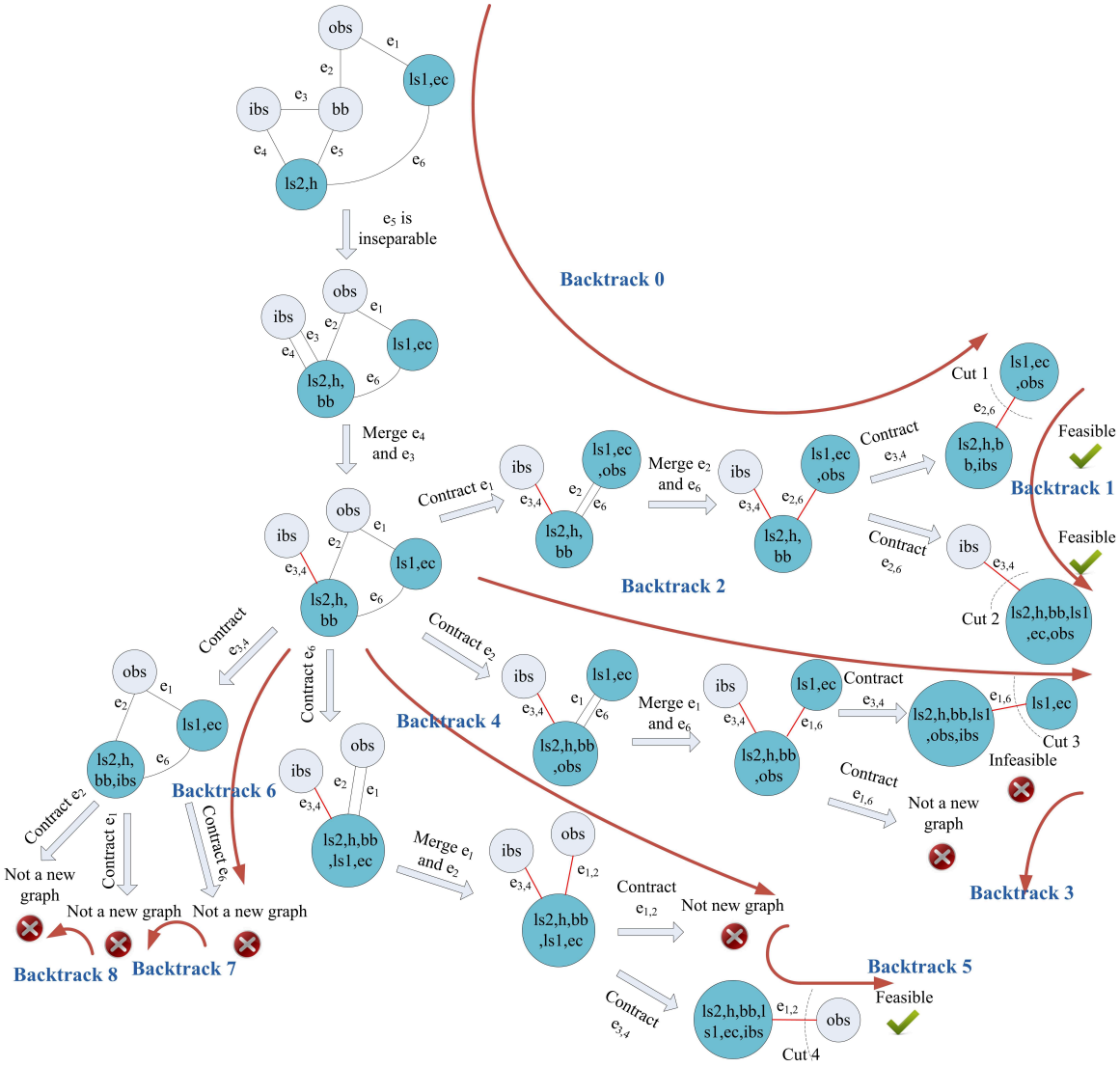


Figure 4-13: Solution space explored by the proposed algorithm for searching the feasible cuts of the bearing assembly example

4.1.5 Summary

Reconfigurable assembly process plan (RAPP) consists of a set of modular assembly process plans, each of which provide the feasible assembly process solutions for a group of interactive product variety components. The interactive product variety components in a product family model are represented at two levels: product family level and product variant level. The modular assembly process plans for the interactive product variety components from product family level are shared by all the product variants in the product family; while the modular assembly process plans for the interactive product variety components from

product variant level are only applicable for a part of specific product variants. With the proposed product family representation model, RAPP can be generated without knowing which product variant in the product family will be manufactured. Once a specific product variant is configured, the feasible assembly process plans can be derived by choosing and combining the corresponding modular assembly process plans from the RAPP.

The DFC model for a product family represents all the mating information between the interactive product variety components of a product family. It consists of a group of connected DFC models at the two variety levels. Each connected DFC model represents the mating relations among the mating parts of a mating module. Each modular assembly process plan in a RAPP corresponds to a mating module. The modular assembly process plans are generated by analysing the connected DFC models in the DFC model. Based on this, the generation of RAPP for a product family is decomposed into the generation of modular assembly process plan for every mating module in the product family.

In order to generate the modular assembly process plan for a mating module, a cut-set based disassembly approach is developed. The approach uses a connected DFC model as an input to analyze the process-related information and the output of this approach is an AND/OR graph which represents the feasible assembly process plans for a mating module. In order to reduce the complexity of the search space and to improve the correctness of the found solution, a pre-process procedure is designed for this approach. It is composed of a series of test-and-mark processes and a reduced graph generation process. In addition, a general algorithm is proposed to realize the cut-set based disassembly approach.

The key process of the cut-set based approach is the generation of all the feasible cuts. For this purpose, a guided Karger's algorithm is proposed. The algorithm adopts the principle of edge contraction in Karger's algorithm but the edges are chosen by the predefined rules instead of randomly chosen edges. By using the guided rules, the solution space can be significantly decreased because these rules can reduce the number of vertices and edges of the graph being analyzed. A new graph judgement mechanism is deployed to guarantee that the reduced graph being explored is always a new reduced graph so as to avoid repeatedly generating the same cut during the execution of the algorithm. Backtrack mechanisms are deployed to guide the algorithm to explore the whole solution space and find all the feasible cuts for the input graph. To illustrate the proposed concepts and algorithms, a bearing assembly is used as an example.

4.2 Generation of reconfigurable machining process plan

Reconfigurable machining process plan (RMPP) provides feasible machining process plans for each part variant in a part family. Based on Definition 30 in chapter 3, RMPP for a part family consists of a set of reconfigurable machining operation plans (RMOPs) and a set of precedence relationships for all the interactive part variety components in the part family. Thus, generation of reconfigurable machining process plan can be divided into two steps:

Step 1: Generate the RMOP for each feature cluster in the part family;

Step 2: Generate the precedence relationships between the interactive part variety components in the part family.

In step 1, the feasible machining operations are selected and sequenced to provide the machining capabilities which satisfy all the machining requirements of the feature cluster. The output of step 1 is a RMOP represented by a directed graph model.

In step 2, the precedence relationships between interactive feature variants are generated by considering a series of factors including feature interaction, setup constraints, tool accessibility and some good practices. In the literature, many approaches for generating the precedence relationships between interactive features have been put forward. Lots of rules and knowledge have been identified for feature sequencing. For example, Ji et al. (2016) define four mapping principles to derive the precedence relationships with considerations of cutting tool and set-up. Mokhtar and Xu (2011) develop a rule-based system to determine the machining precedence of interactive features. Liu and Wang (2007)define a set of rules to determine feature precedence with considerations of fixture interactions, tool interactions and tolerance/datum interactions. Sormaz and Khoshnevis (2000) propose an algorithm to generate a feature precedence network for a given part considering the geometric, technological and economical factors on feature precedence.

Therefore, in this thesis work, it is assumed that the precedence relationships between interactive feature variants are already given. The precedence relationships can be represented at both feature cluster level and feature variant level according to the rules defined in chapter 3. The given precedence relationships are then used to configure the machining process plans for a specific part variant in chapter 5. This work only focuses on the step 1 that is generation of RMOP for a feature cluster.

4.2.1 Generation of RMOP for a feature cluster

A RMOP consists of a set of similar machining operation sequences sharing a part of common machining operations and a part of the same precedence between the common operations. These similar machining operation sequences provide the necessary machining capabilities to satisfy all the machining requirements of a feature cluster. The machining requirements of a feature cluster correspond to the process-related design specifications of all the feature variants in this feature cluster, whose values are represented as intervals or sets.

Because different machining operations have different machining capabilities and different types of feature clusters have different design specifications, selecting an appropriate machining operation for an appropriate feature cluster is one of the issues that have to be addressed when generating the RMOP.

In addition, when more than one appropriate machining operations are selected for a feature cluster, how to sequence these machining operations to satisfy the machining requirements of the feature cluster as well as to derive the alternative machining operation routes is another issue that have to be solved for the generation of RMOP.

To address the above issues and generate the RMOP for a feature cluster, a two-steps approach is proposed in this section. Figure 4-14 shows the proposed approach to generate a RMOP for a feature cluster(Xia et al., 2016).

In this approach, the process-related design specifications of a feature cluster are decomposed into three classes:

- Geometric specification. The specification for the geometric attributes inherited from a form feature. For example, a hole cluster could have a diameter specification and a depth specification.

- Tolerance specification. The specification for the dimensional tolerances and geometrical tolerances of the geometric elements on the feature variants of a feature cluster. For example, a hole cluster could have dimensional tolerance constraints on the diameters of its variants or roundness constraints on the cylindrical surfaces of its variants.

- Surface finish specification. The specification for the surface roughness or hardness of the important surfaces of the feature variants in a feature cluster.

Accordingly, the capabilities of a machining operation are also classified into:

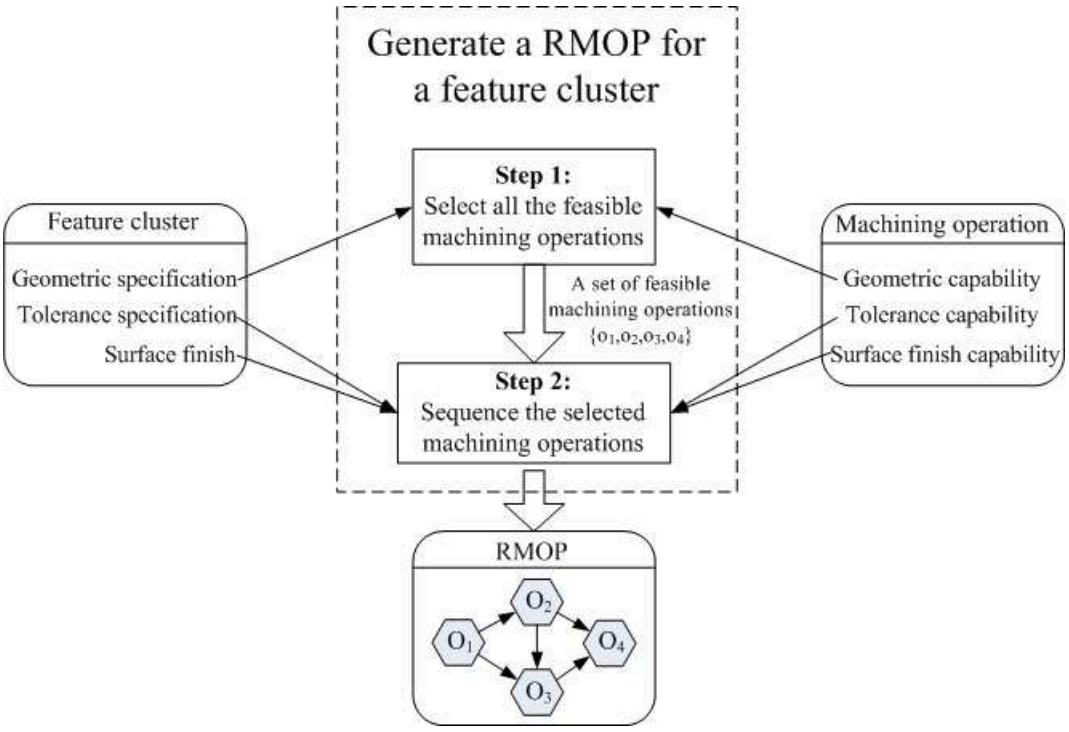


Figure 4-14: Proposed approach to generate a RMOP for a feature cluster

- Geometric capability. The capabilities to generate a geometric shape and volume on a workpiece.
- Tolerance capability. The machining quality and accuracy of a machining operation.
- Surface finish capability. The surface roughness or hardness of a surface after applying a machining operation.

The principle of the proposed approach is to match the machining capabilities of machining operations with the corresponding machining requirements of a feature cluster. The matching process is divided into two steps:

- In the first step, all the feasible machining operations are firstly chosen from a set of machining operation candidates according to their geometric compatibilities compared with the geometric specification of the feature cluster. First order logic language is used to represent the knowledge on the geometric capabilities of machining operations. A knowledge base is designed to structure the representation of operation selection knowledge. In addition, a Resolution-based Breadth-First Algorithm (R-BF algorithm) is proposed to select all the geometrically compatible machining operations

with a feature cluster.

- After selecting all the feasible machining operations, in the second step, these feasible machining operations are sequenced according their tolerance capabilities and surface finish capabilities to satisfy the corresponding tolerance specifications and surface finish specifications of the feature cluster. A mathematical model is proposed to model the machining operation sequencing problem and a depth-first algorithm is proposed to solve this problem.

Select all the feasible machining operations

In production practice, the machining operation candidates could have different types, such as turning operations, milling operations, drilling operations and grinding operations. Each type has its specific machining capabilities and scope of application, for example, turning operations can only be applied on a revolving workpiece. In addition, even for the operations with the same process type, they could have different machining capabilities if their machine tools and cutting tools are different. Therefore, in order to automatically choose the feasible machining operations for a feature cluster, the system must have the ability to represent the knowledge on the geometric capabilities of different machining operations and the ability to infer the feasible machining operations for a specific feature from the knowledge.

A first-order-logic-based knowledge base for machining operation selection is developed. The knowledge base organizes the basic domain knowledge for geometric capabilities of different machining operations and the knowledge for operation selection. The knowledge in this knowledge base is expressed by a logic language - First Order Logic (FOL).

Figure 4-15 shows the structure of this knowledge base. The knowledge base is built upon a set of facts that describe the domain knowledge. The facts are built upon a basic vocabulary which consists of five classes of domain-dependent knowledge elements:

- **Individual**. An individual represents a real entity in the domain. It is an instance of an object class involving in the domain of RMOP generation. In FOL, it is represented by a constant symbol, like milling1, drilling1, holeCluster1, normalBoring1.

- **Type**. A type describes an object class to which those individuals belong. In FOL, it is represented by an unary predicate, such as *milling*(milling1), *drilling*(drilling1), *hole*(holeCluster1).

- **Attribute**. An attribute describes a property or a status that can hold for individuals. In FOL, it is also in the form of an unary predicate, such as *roughProcess*(drilling1), *finishProcess*(grinding1), *closed*(pocket1).

- **Relationship**. A relationship describes a relationship between n individuals. In FOL, a n-predicate is used to express a relationship, such as *geometricallyCompatible*(drilling1, holeCluster1).

- **Function**. A function describes a mapping relationship from a domain to a range. In FOL, a function is represented by a n-predicate with the first letter capitalized, such as *Diameter*(holeCluster1), *MaxToolDiameter*(drilling1).

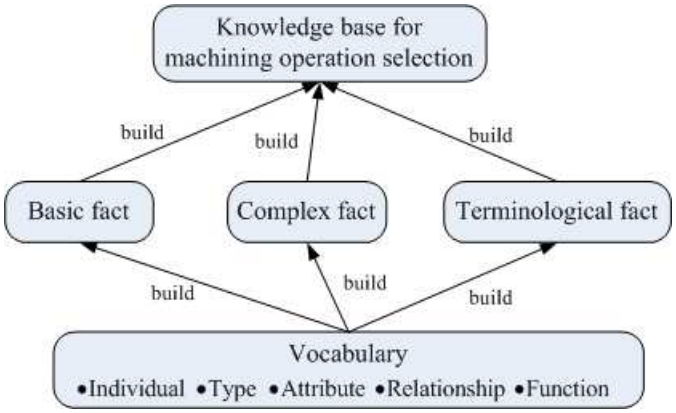


Figure 4-15: Structure of a FOL Knowledge base for generation of a RMOP

After defining the vocabulary of the knowledge base, these basic knowledge representation elements are used to build the facts. There are three types of facts in the knowledge base:

- **Basic fact**. A basic fact is the simplest fact in the knowledge base and it is represented by atomic formulas and negation of atomic formula. For example, *drilling*(drilling1), *geometricallyCompatible*(turning1, holeCluster1).

- **Complex fact**. Besides the basic facts, more complex facts should be also expressed in the domain. In a complex fact, the basic facts are constrained by quantifiers $(\forall, \exists)$ and connected by logic connectives $(\wedge, \vee, \leftarrow, \rightarrow, \leftrightarrow)$ to form a complex logic formulas. For example, to express a fact that all the drilling operations are geometrically compatible with a round hole, we can write a logic sentence as follow:

$$\forall x, y(drilling(x) \wedge roundHole(y) \rightarrow geometricallyCompatible(x, y)).$$

- **Terminological fact**. Besides expressing the relationships among individuals, a robust knowledge base should also include the relationships among predicates and functions. A terminological fact describes the relationship among the predicates and function symbols. Brachman and Levesque (2004) identify seven types of relationships between predicates and functions: disjointness, subsumption, exhaustiveness, symmetry, inverse, type restriction and full definition. For example,
  - Disjointness:

$$\forall x, y(geometricallyCompatible(x, y) \leftrightarrow \neg geometricallyIncompatible(x, y));$$

  - Subsumption:

$$\forall x(twisDrilling(x) \rightarrow drilling(x));$$

  - Exhaustiveness:

$$\forall x(hole(x) \rightarrow through(x) \vee open(x));$$

  - Symmetry:

$$\forall x, y(equal(x, y) \leftrightarrow equal(y, x));$$

  - Inverse:

$$\forall x, y(larger(x, y) \leftrightarrow less(y, x));$$

  - Type restriction:

$$\begin{aligned} \forall x, y(compatible(x, y) \quad \leftrightarrow \quad & (operation(x) \vee operationSequence(x)) \wedge \\ & featureCluster(y)). \end{aligned}$$

  - Full definition:

$$\begin{aligned} \forall x, y(compatible(x, y) \quad \leftrightarrow \quad & geometricallyCompatible(x, y) \wedge \\ & toleranceCompatible(x, y) \wedge \\ & surfaceFinishCompatible(x, y)). \end{aligned}$$

Table 4.1: Vocabulary of a simple knowledge base for hole-making process selection

Vocabulary type	Literal	Meaning
Individual	hc1	An individual named hc1
	td	An individual named td
	r	An individual named r
	t	An individual named t
Type	$h(x)$	The feature type of x is a hole
	$t(x)$	The process type of x is turning
	$td(x)$	The process type of x is twist drilling
	$r(x)$	The process type of x is reaming
Attribute	$hm(x)$	x is a hole-making operation
Relationship	$gc(x, y)$	Machining operation x is geometrically compatible with feature cluster y
	$tdc(x, y)$	The diameter of x's cutting tool is compatible with feature cluster y
	$tlc(x, y)$	The length of x's cutting tool is compatible with feature cluster y
	$leoe(x, y)$	x is less or equal than y
	$laoe(x, y)$	x is large or equal than y
Function	$D(x)$	A function outputs the diameter of x
	$MATD(x)$	A function outputs the maximum tool diameter of x
	$MITD(x)$	A function outputs the minimum tool diameter of x
	$DDR(x)$	A function outputs the ratio of depth to diameter of x
	$LDRL(x)$	A function outputs the limitation of the ratio of length to diameter of x's cutting tool

In order to illustrate the function of the proposed knowledge base for machining operation selection, we build a simple knowledge base for selecting a hole-making operation for a hole cluster. The vocabulary of this simple knowledge base is listed in table 4.1.

After defining the vocabulary for our knowledge base, we built the following facts to represent the knowledge for hole-making operation selection:

1. For all hole making operations and all holes, if the operation is tool diameter compatible and tool length compatible with the hole, then the operation is geometrically compatible with the hole:

$$\forall x, y(hm(x) \wedge h(y) \wedge tdc(x, y) \wedge tlc(x, y) \rightarrow gc(x, y));$$

2. For all hole making operations and all holes, if the diameter of the hole is less or equal than the maximum tool diameter of the operation and larger or equal than

the minimum tool diameter of the operation, then the operation is tool diameter compatible with the hole:

$$\forall x, y(hm(x) \wedge h(y) \wedge leoe(D(y), MATD(x)) \wedge laoe(D(y), MITD(x)) \rightarrow tdc(x, y));$$

3. For all hole making operations and all holes, if the hole's ratio of depth to diameter is less or equal than the limitation of the ratio of length to diameter of the operation's cutting tool:

$$\forall x, y(hm(x) \wedge h(y) \wedge leoe(DDR(y), LDRL(x)) \rightarrow tlc(x, y));$$

4. A twist drilling operation is a hole making operation: $\forall x(td(x) \rightarrow hm(x))$;

5. A reaming operation is a hole making operation: $\forall x[r(x) \rightarrow hm(x)]$;

6. A turning operation is not a hole making operation: $\forall x(t(x) \rightarrow \neg hm(x))$;

7. td is a twist drilling operation: td(td);

8. r is a reaming operation: r(r);

9. t is a turning operation: t(t);

10. hc1 is a hole feature cluster: h(hc1);

11. The diameter of hc1 is less or equal than the maximum diameter of td's cutting tool:

$$leoe(D(\text{hc1}), MATD(\text{td}));$$

12. The diameter of hc1 is less or equal than the maximum diameter of r's cutting tool:

$$leoe(D(\text{hc1}), MATD(\text{r}));$$

13. The diameter of hc1 is larger or equal than the minimum diameter of td's cutting tool:

$$laoe(D(\text{hc1}), MITD(\text{td}));$$

14. The diameter of hc1 is larger or equal than the minimum diameter of r's cutting tool:

$$laoe(D(\text{hc1}), MITD(\text{r}));$$

15. The ratio of depth to diameter of hc1 is less or equal than limitation of the ratio of length to diameter of td's cutting tool:

$$leoe(DDR(\text{hc1}),LDRL(\text{td}));$$

16. The ratio of depth to diameter of hc1 is less or equal than limitation of the ratio of length to diameter of r's cutting tool:

$$leoe(DDR(\text{hc1}),LDRL(\text{r})).$$

The 16 pieces of facts compose the knowledge for our simple knowledge base. In order to reason the knowledge base to select the feasible machining operations for a feature cluster, we propose a R-BF algorithm. The algorithm receives a query and then tries to find the answer for this query through reasoning the facts in the knowledge base. In natural language, the query could be written as: "are there any feasible machining operations for machining the feature cluster? If yes, then what are they?". The query could be transformed into a FOL formula like:$\exists(gc(x,\text{a}) \wedge \neg A(x))$, where $gc(x,\text{a})$ is a relationship defined in table 4.1; $A(x)$ is a predicate occurring nowhere else and it is used to extract the answer of this query, called answer predicate.

To reason this query, the algorithm adopts a resolution mechanism which can judge whether the knowledge base entails the query and then extract the answer from the reasoning result.

The resolution mechanism is established on the clausal forms of Conjunctive Normal Form (CNF), thus the facts in the knowledge base and the query are firstly converted into the clausal forms of CNF. A formula is in CNF if it is a conjunction of clauses, where a clause is a disjunction of atomic formulas. For example: $\forall x, y((p(x) \vee r(\text{a})) \wedge q(x,y) \wedge (s(y) \vee t(\text{b})))$ is in CNF.

The clausal form of CNF is using "[" and "]" as the delimiters for clauses and using "{" and "}" as the delimiters for a clausal formula. Any FOL formula can be converted into a clausal form of CNF by following the approach proposed by (Brachman and Levesque, 2004). For instance, the CNF formula in the previous example can be converted into a clausal form like:

$$\{[p(x), r(\text{a})], [q(x,y)], [s(y), t(\text{b})]\}.$$

Therefore, the facts in our knowledge base for hole-making operation selection can be converted into the following clausal forms of CNF:

$F_1 = [\neg hm(x), \neg h(y), \neg tdc(x,y), \neg tlc(x,y), gc(x,y)]$
$F_2 = [\neg hm(x), \neg h(y), \neg leoe(D(y), MATD(x)), \neg laoe(D(y), MITD(x)), tdc(x,y)]$
$F_3 = [\neg hm(x), \neg h(y), \neg leoe(DDR(y), LDRL(x)), tlc(x,y)]$
$F_4 = [\neg td(x), hm(x)]$
$F_5 = [\neg r(x), hm(x)]$
$F_6 = [\neg t(x), hm(x)]$
$F_7 = [td(\mathrm{td})]$
$F_8 = [r(\mathrm{r})]$
$F_9 = [t(\mathrm{t})]$
$F_{10} = [h(\mathrm{hc1})]$
$F_{11} = [leoe(D(\mathrm{hc1}), MATD(\mathrm{td}))]$
$F_{12} = [leoe(D(\mathrm{hc1}), MATD(\mathrm{r}))]$
$F_{13} = [laoe(D(\mathrm{hc1}), MITD(\mathrm{td}))]$
$F_{14} = [laoe(D(\mathrm{hc1}), MITD(\mathrm{r}))]$
$F_{15} = [leoe(DDR(\mathrm{hc1}), LDRL(\mathrm{td}))]$
$F_{16} = [leoe(DDR(\mathrm{hc1}), LDRL(\mathrm{r}))]$

The principle of the resolution mechanism is based on the following theorem:

Theorem 1 (Ben-Ari (2012)). *$U \models \alpha$ if and only if $U \cup \{\neg\alpha\}$ is unsatisfiable where U is a set of FOL formulas and α is a FOL formula.*

Therefore, the aim of resolution mechanism is to find whether the set of formulas: KB$\cup\{\neg q\}$ is unsatisfiable or not. KB is the knowledge base and $\neg q$ is the negation of the query. If KB$\cup\{\neg q\}$ is unsatisfiable, then there exists a feasible machining operation in the knowledge base and the answer predicate $A(x)$ will deliver the feasible machining operation.

To reason the satisfiability of KB$\cup\{\neg q\}$, the resolution mechanism uses a resolution rule to resolve the clauses in KB$\cup\{\neg q\}$ until a clause that only contain the answer predicate $A(x)$ is derived or no clause can be resolved. The resolution rule has the following form:

$$C = (C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma),$$

where C_1 and C_2 are two clauses in KB$\cup\{\neg q\}$; $L_1 = \{l_1^1, ..., l_n^1\} \subseteq C_1$ and $L_2 = \{l_1^2, ..., l_n^2\} \subseteq C_2$; σ is a most general unifier (MGU) which unifies L_1 and $L_2^c = \{\neg l_1^2, ..., \neg l_n^2\}$; C is a resolvent after applying the resolution rule. L_1 and L_2^c are said to be unified if there is a substitution θ such that $L_1\theta = L_2^c\theta$. A unifier σ is a MGU if any other unifier θ can be derived by applying a further substitution λ on σ, namely $\theta = \sigma\lambda$. A MGU of a pair of

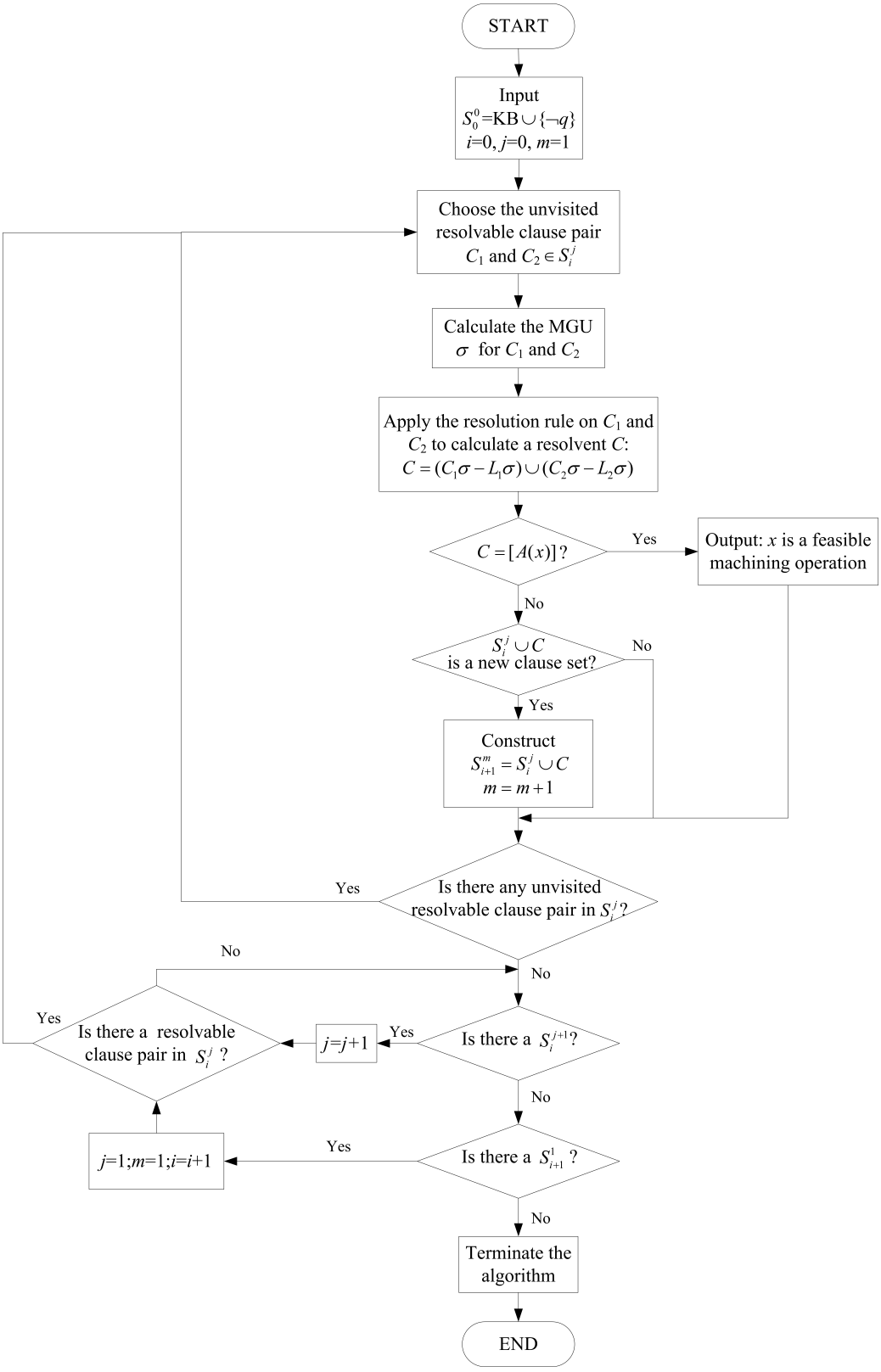


Figure 4-16: Flowchart of a R-BF algorithm for machining operation selection

literal can be calculated efficiently by a procedure proposed by Brachman and Levesque (2004).

With the resolution mechanism, the reasoning algorithm can find out a feasible machining operation if there exists one in the knowledge base. But the resolution mechanism cannot guarantee finding all the feasible operations for a feature cluster. Therefore, for this purpose, a breadth-first searching mechanism is developed to explore the solution space of the searching tree. The breadth-first searching mechanism not only can guarantee finding all the feasible operations, but also can prevent that the algorithm loops in an infinite searching branch.

Figure 4-16 shows the flowchart of the proposed R-BF algorithm. At the beginning, a clause set S_0^0 combined from the CNF's clausal form of KB and the corresponding form of the negation of the query $\{\neg q\}$ is input into the algorithm as well as three counters, $i = 0$, $j = 0$ and $m = 1$. Then, two unvisited resolvable clauses C_1 and C_2 is chosen from S_i^j and the MGU σ of C_1 and C_2 is calculated; after that, the resolution rule is applied on C_1 and C_2 to derive a resolvent C. If C is a clause only containing the answer predicate $A(x)$, then a feasible machining operation is found and the algorithm will output the operation, then it continues to find next unvisited resolvable clause pair in S_i^j. If C contains some other predicates except $A(x)$ and $S_i^j \cup C$ is a new clause set, then construct a new clause set as $S_{i+1}^m = S_i^j \cup C$. The algorithm repeats the above processes to construct all the possible clause sets for S_i^j until there is no unvisited resolvable clause pair in S_i^j.

After constructing all the possible clause sets for S_i^j, the algorithm checks whether there is a S_i^{j+1}, if yes, then repeating the above processes to construct all the possible clause sets for S_i^{j+1}, otherwise, the algorithm checks whether there is a S_{i+1}^1, if yes, then the algorithm assigns new values to the counters: $j = 1$,$m = 1$, $i = i + 1$, and checks whether there is a resolvable clause pair in the new clause set S_i^j, if yes, then it repeats the processes to construct all the possible clause sets for S_i^j, otherwise, it repeats the processes to check whether there is a next clause set S_i^{j+1}. The algorithm terminates until there is no S_{i+1}^1 can be found in the searching tree.

Figure 4-17 shows a part of the searching process of the R-BF algorithm. In this searching process, two feasible machining operations, td and r, are found according the knowledge built in our simple knowledge base. Turning operation t is discarded because the literal "hm(t)" is not resolvable in the knowledge base. The R-BF algorithm is independent of the knowledge

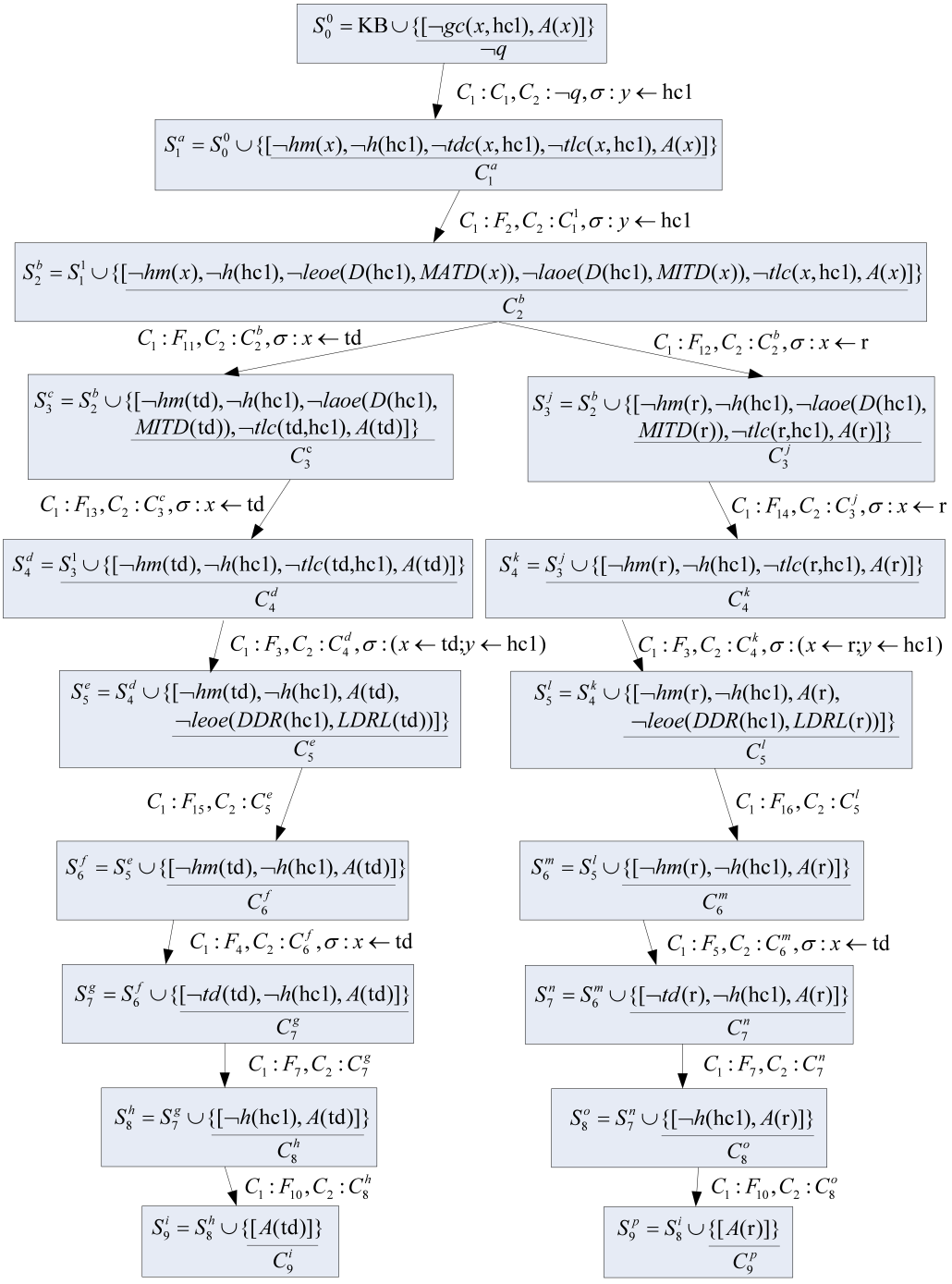


Figure 4-17: Partial searching process of the proposed R-BF algorithm

base so that it can be apply to any FOL knowledge base with different domain knowledge for operation selection.

Sequence the selected machining operations

After step 1, the geometrically feasible machining operations for a feature cluster are found. In step 2, the possible sequences of these machining operations should be arranged according to their tolerance capabilities and surface finish capabilities. In order to better describe this sequencing problem, a mathematical model is built with the following reasonable assumptions:

- The considered machining operations are geometrically compatible with the feature cluster. Therefore, in this step, only tolerance capabilities and surface finish capabilities of these machining operations are considered.
- The tolerance specifications and the surface finish specifications of a feature cluster are represented by the means of intervals or sets; this assumption is in conformity with the proposed representation model of feature cluster in chapter 2.
- The tolerance capabilities and the surface finish capabilities of a machining operation are also represented by the means of intervals or sets. This assumption is reasonable because the capabilities of a machining operation can vary in a range depending on its cutting tool, machine tool and machining parameters.

Based on the above assumptions, we define the following notions:

fc : a feature cluster;

o_i : a machining operation, $i = 1, 2, ..., n$;

op_i : a machining operation plan, $i = 1, 2, ..., n$;

r_j^{fc} : a machining requirement of a feature cluster fc on j; j is a type of tolerance specifications or surface finish specifications, $r_j^{fc} = [min(r_j^{fc}), max(r_j^{fc})]$;

$c_j^{o_i}$: a machining capability of an operation o_i on j; $c_j^{o_i} = [min(c_j^{o_i}), max(c_j^{o_i})]$;

$c_j^{op_i}$: a machining capability of an operation plan op_i on j; $c_j^{op_i} = [min(c_j^{op_i}), max(c_j^{op_i})]$.

With the above notions, we can define the sequencing problem as:

Given:

1. A set of machining requirements $R = \{r_{j_1}^{fc}, r_{j_2}^{fc}, ..., r_{j_n}^{fc}\}$ of a feature cluster fc;
2. A set of machining operations $O = \{o_1, o_2, ..., o_m\}$ and each machining operation $o_k, k = 1, 2, ..., m$, has a set of machining capabilities $C^{o_k} = \{c_{j_r}^{o_k} | r \in 1, 2, ..., n\}$.

Goal:

To find all the possible machining operation plans $op_i = \{(o_s, o_t)|o_s \neq o_t; o_s, o_t \in O\}$ such that all the machining requirements$\in R$ of the feature cluster fc are satisfied respectively by the corresponding machining capabilities $C^{op_i} = \{c_{j_r}^{op_i}|r = 1, 2, ..., n\}$ of op_i.

In order to solve this problem, we define the following operation sequencing knowledge:

- The capabilities C^{op_i} of a machining operation plan op_i depend on the last machining operation in the operation sequence:

$$C^{op_i} = \{c_{j_r}^{op_i}|c_{j_r}^{op_i} = c_{j_r}^{o_n} \wedge op_i = \{(o_1, o_2), (o_2, o_3), ..., (o_{n-1}, o_n)\} \wedge r = 1, 2, ..., n\}.$$

- In a machining operation plan op_i, the latter operation should always improve the machining capabilities delivered by the machining sequence before it:

$$\forall o \in op_i, \exists j(sequenceBefore(op_k, o) \rightarrow min(c_j^o) < min(c_j^{op_k})).$$

- A machining operation plan op_i satisfies a feature cluster fc, if and only if for all the machining requirements, there exists one capability of op_i whose minimum value is less or equal than the minimum value of the corresponding machining requirement:

$$satisfy(op_i, fc) \rightarrow \forall r_{j_k}^{fc} \in R, \exists c_{j_k}^{op_i} \in C^{op_i}(min(c_{j_k}^{op_i}) \leq min(r_{j_k}^{fc})).$$

- Rough operations should always be executed before semi-finishing operations and semi-finishing operations should always be executed before finishing operations. In addition, finishing operations should not be arranged at the beginning of a machining operation plan.

Based on the above knowledge for operation selection, an operation sequencing algorithm is proposed. The flowchart of this algorithm is shown in figure 4-18. The inputs of this algorithm consist of the machining requirements of a feature cluster and a set of feasible machining operations as well as their machining capabilities. The algorithm outputs all the possible machining operation plans, each of which satisfies all the manufacturing requirements of the feature cluster. When one feature variant is instantiated from the feature cluster, one of these machining operation plans or a partial plan tailored according to the machining requirements of this feature variant can be chosen.

All the possible machining operation plans generated from the algorithm compose a RMOP for the feature cluster. In the algorithm, the machining operation plans are represented by a special data structure:$\{(\text{st}, o_1), (o_1, o_2), ..., (o_{n-1}, o_n)\}$, where st is a start point of the operation plan and o_n is a machining operation from the operation set, (o_{n-1}, o_n) describes that operation o_{n-1} precedes operation o_n. By using such a data structure, the directed graph of the RMOP for the feature cluster is derived explicitly: $o_1, o_2, ..., o_n$ are the nodes in the directed graph and (o_{n-1}, o_n) are the edges.

The knowledge defined above is used to help the algorithm identify the right sequence between machining operations. At the beginning, one machining operation is randomly chosen from the operation set, if it can be the first operation (not a finishing operation or other special operations), then a machining operation plan only containing this operation is constructed.

After that, its machining capabilities are calculated and compared with the requirements; if it satisfies all the requirements, then output this machining operation plan and take another unvisited operation to construct a new machining operation plan, otherwise, the algorithm expands this first plan by finding another unvisited machining operations.

The machining operation being added to the plan must improve the capability of current machining operation plan and it must be compatible with the last operation in current machining operation plan, otherwise, that machining operation is discarded and marked as visited in the operation candidate set for current plan.

In the operation candidate set of an operation plan, if an operation candidate is visited, then it is marked as visited. The algorithm continues to expand a machining operation plan until one is found that its capabilities satisfy all the machining requirements of a feature cluster or there is no unvisited operation candidate for this machining operation plan. If a right machining operation plan is found or there is no unvisited operation candidate to be expanded, then the algorithm backtracks to nearest unvisited operation candidate and it continues to expand the corresponding machining operation plan. When it backtracks to the level of op_0 and all the operation candidates are visited, then it terminates.

This algorithm has been applied to generate a RMOP for a hole cluster - *HC2* of the oil pump body family example. The machining requirements of $HC2$ are given in table 3.1 in the chapter 3. Four hole-making operations are input as the feasible machining operation for *HC2* and the machining capabilities of these operations are listed in table 3.2. In addition,

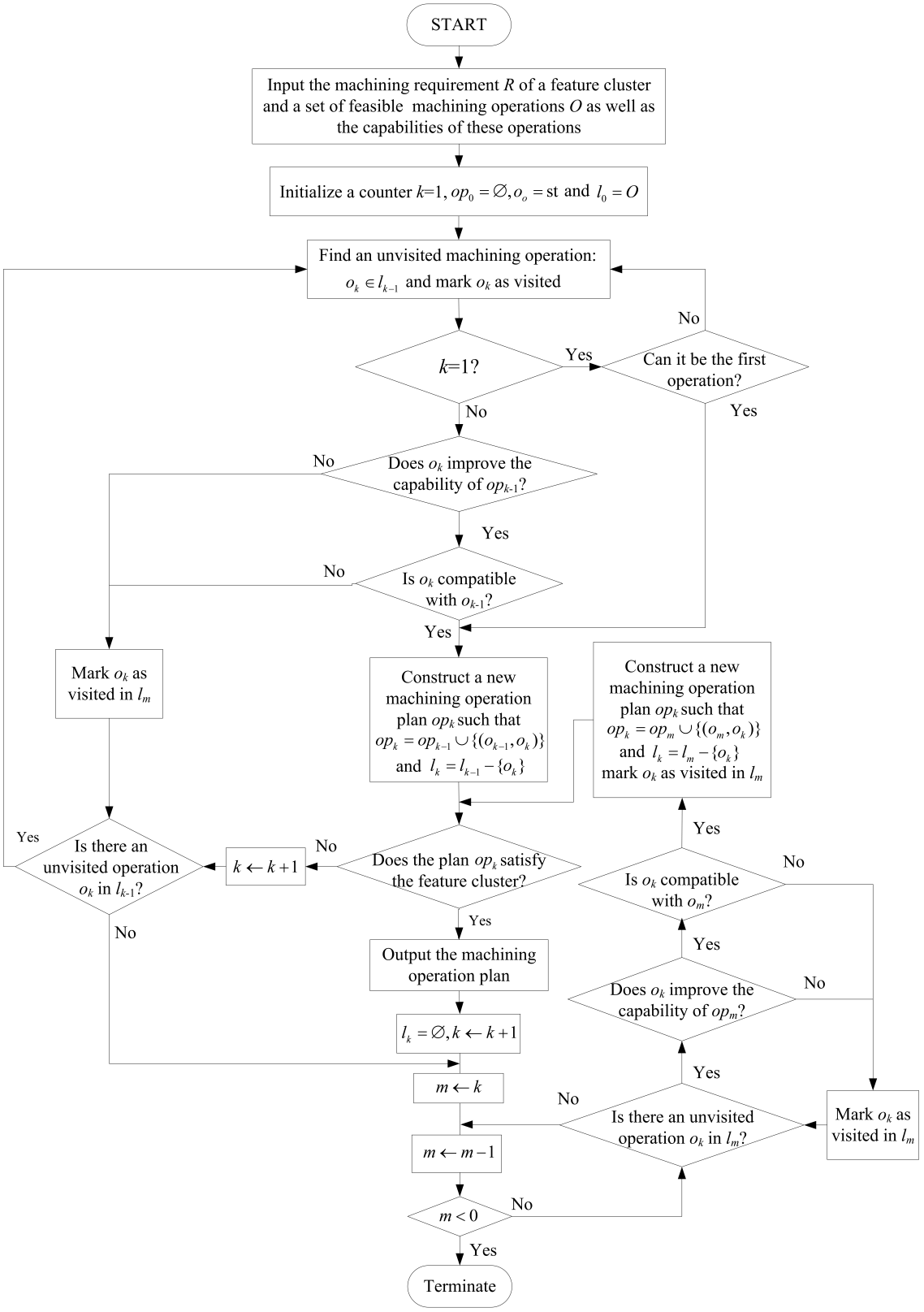


Figure 4-18: Flowchart for the proposed machining operation sequencing algorithm

we have the following machining knowledge:

- Twist drilling o_{td} is a rough operation;
- Reaming o_r is a semi-finishing operation;
- Normal boring o_{nb} is a semi-finishing operation;
- o_r and o_{nb} cannot be the first machining operations in a machining operation plan;
- Precision boring o_{pb} is a finishing operation.

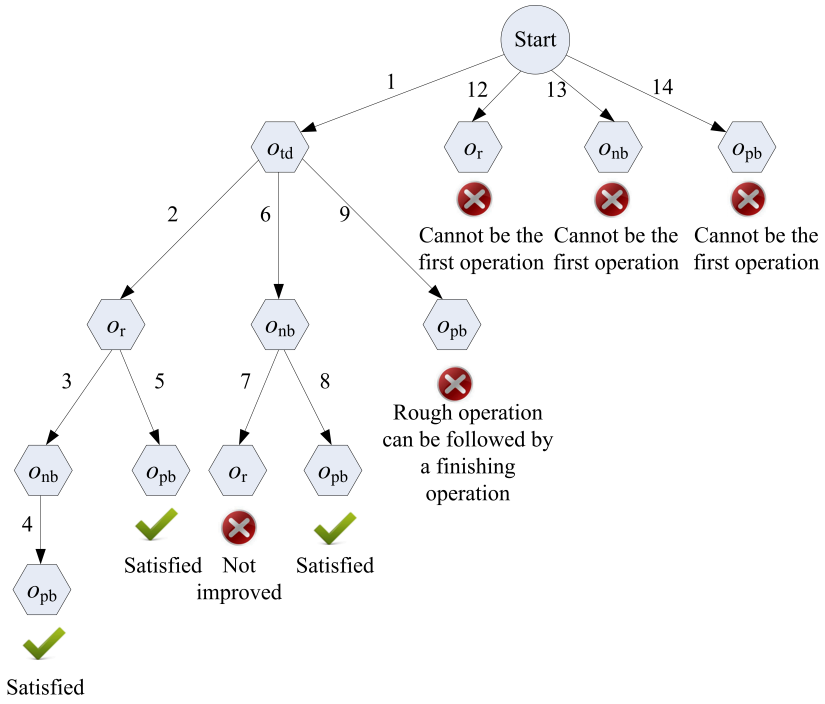


Figure 4-19: Searching tree of the proposed algorithm to generate the RMOP of a hole cluster: HC2

Figure 4-19 shows the searching tree of the proposed algorithm to arrange the sequences among the four hole-making operations. The numbers on the arrows in the tree mark the orders of expanding a machining operation plan by the algorithm. At the beginning, the operation candidate set is $l_0 = \{o_{td}, o_r, o_{nb}, o_{pb}\}$, and then o_{td} is chosen randomly as the first machining operation and marked as visited in l_0, the operation sequence $op_1 = \{(\text{st}, o_{td})\}$ only satisfy the true position requirement of $HC2$ $(min(c_{tp}^{o_{td}}) < min(r_{tp}))$, therefore, it is expanded by the algorithm, the operation candidate set for op_1 is $l_1 = \{o_r, o_{nb}, o_{pb}\}$.

From the candidate set, o_r is randomly chosen as the potential subsequence of op_1 and marked as visited in l_1. As o_r is a semi-finishing operation and it improves the capabilities of op_1 on dimensional tolerance, surface roughness and cylindricity, op_1 is expanded by o_r which results a new operation plan $op_2 = \{(\text{st}, o_{td}), (o_{td}, o_r)\}$ and op_2 further improves the machining capabilities of the sequence and satisfies the surface roughness requirement of *HC2* $(min(c_{sr}^{o_r}) < min(r_{sr}))$.

Because not all the requirements are satisfied and there is an unvisited operation candidate in the candidate set $l_2 = \{o_{nb}, o_{pb}\}$ for op_2, the algorithm continues to expand op_2 by randomly choosing an operation from the candidate set. The expanding process is repeatedly executed until an operation plan $op_4 = \{(\text{st}, o_{td}), (o_{td}, o_r), (o_r, o_{nb}), (o_{nb}, o_{pb})\}$, at this time, all the requirements are satisfied by the capabilities of op_4, therefore, the first machining operation plan for $HC2$ is op_4.

In the next steps, the algorithm backtracks to the nearest point where there is an unvisited operation candidate for expanding that is op_2 with an unvisited operation candidate set $\{o_{pb}\}$, and then the second machining operation plan is found that is $op_5 = \{(\text{st}, o_{td}), (o_{td}, o_r), (o_r, o_{pb})\}$. After outputting op_5, the operation candidate set l_5 for op_5 is set to an empty set because it is unnecessary to expand a machining operation plan that has satisfied all the requirements.

After that, the algorithm backtracks to the next unvisited operation o_{nb} in l_1 and uses this operation to expand op_1; as a result, $op_6 = \{(\text{st}, o_{td}), (o_{td}, o_{nb})\}$ and $l_6 = \{o_r, o_{pb}\}$ are derived and then o_r is tried for op_6, but the algorithm find it does not improve the capabilities of op_6, so another unvisited operation o_{pb} in l_6 are added onto op_6 and the third machining operation plan $op_7 = \{(\text{st}, o_{td}), (o_{td}, o_{nb}), (o_{nb}, o_{pb})\}$ which satisfies all the requirements is derived. In the next searching process, the algorithm traverses all the rest of unvisited operations and no new targeted machining operation plan is found.

In this example, the algorithm finds three machining operation plans that can satisfy all the machining requirements of the hole cluster $HC2$. These three machining operation plans compose a RMOP for the hole cluster $HC2$ as illustrated in figure 4-20. The machining operation plans for any hole variant in this hole cluster can be derived immediately from the RMOP instead of re-generating them from a scratch. For instance, if a hole variant has the machining requirements: $r_{dt} = 8, r_{sr} = 1.6, r_{tp} = 0.04, r_{cy} = 0.05$, then three machining operation plans can be identified from RMOP by applying a route searching algorithm on

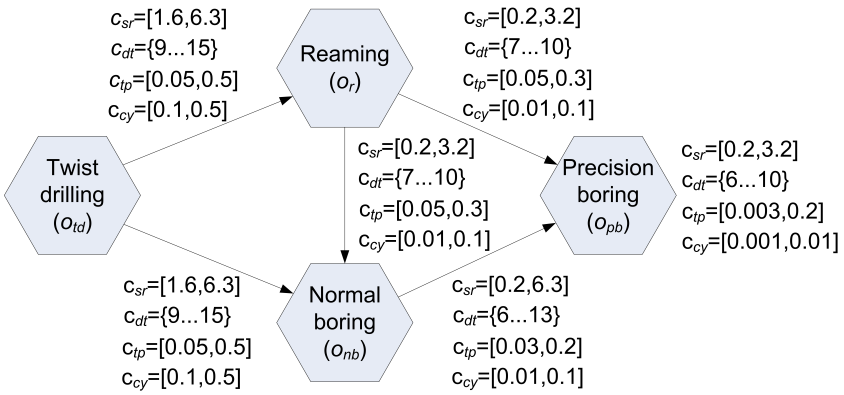


Figure 4-20: RMOP for a hole cluster: HC2

the directed graph: $\{(o_{td}, o_r), (o_r, o_{pb})\}, \{(o_{td}, o_{nb})\}$ and $\{(o_{td}, o_r), (o_r, o_{nb})\}$.

4.2.2 Summary

RMPP for a part family consists of a set of RMOPs for the feature clusters and a set of precedence relationships for the interactive part variety components in the part family. Consequently, generation of RMPP includes two steps: the first step is to generate the RMOPs for each feature cluster; the second step is to generate the precedence relationships between the interactive part variety components. This section focuses on the first step. A two-steps approach is proposed for the generation of RMOP; in step 1, the approach selects all the feasible machining operations which are geometrically compatible with a given feature cluster, then in step 2, the selected machining operations are sequenced to construct a RMOP according to their machining process capabilities and the machining requirements of the feature cluster.

For the step 1 in the two-steps approach, a FOL-based knowledge base is proposed to organize the domain knowledge on machining operation selection and then a R-BF algorithm is proposed to reason the facts in the knowledge base for searching the feasible machining operations. The resolution mechanism of the proposed R-BF algorithm can guarantee finding a feasible operation if there is one. The breadth-first mechanism can guarantee finding all the feasible operations and preventing an infinite searching loop. A simple FOL knowledge base for hole-making operation selection is built and the reasoning process of the proposed algorithm applied on this knowledge base is illustrated to show the feasibility and effectiveness of the proposed approach.

For the step 2, a mathematical model is defined for the machining operation sequencing problem. In order to solve the sequencing problem, the knowledge necessary for the problem solving is defined and a depth-first operation sequencing algorithm is proposed to explore the whole possible solution space. The algorithm has been applied to generate a RMOP for a hole cluster - $HC2$ of the oil pump body family example to show the feasibility and effectiveness.

4.3 Conclusions

This chapter deals with the generation of RPP from two levels of granularity. At the level of product family, the generation of a RAPP is decomposed into the generations of a set of modular assembly process plans for the interactive product variety components in the product family. A cut-set based disassembly approach is proposed to generate the modular assembly process plans through analyzing the process-related information represented by the connected DFC model of a mating module. The output of this approach is an AND/OR graph representing the modular assembly process plans for a mating module. The AND/OR graphs of all the modular assembly process plans in a product family compose the final RAPP.

At the level of part family, the generation of RMPP is divided into two steps: generation of RMOPs and generation of feature precedence. Because there are many approaches available in the literature for the generation of feature precedence, in this research, we only focus on the approach to generate a RMOP for a feature cluster. For this purpose, a two-steps approach is proposed: all the feasible machining operations are selected in the first step, and then in the second step the selected operations are sequenced to generate the RMOP for a feature cluster. A knowledge based reasoning algorithm is proposed for the first step, this algorithm uses a resolution mechanism to reason the domain knowledge on machining operation selection represented in a FOL-based knowledge base. A breadth-first reasoning mechanism is also deployed in the algorithm to explore the whole possible solutions in the solution space. In the second step, in order to solve the machining operation sequencing problem, a depth-first based algorithm is proposed. The machining operation plans generated by this algorithm explicitly form the RMOP for a feature cluster.

The proposed product variety representation model and the part variety representation

model are used to provide the process-related information for RPP generation and the outputs of the algorithms for RPP generation are in accordance with the RPP models proposed in chapter 3. Examples from the gear pump family and the oil pump body family are used to illustrate the feasibility and effectiveness of the proposed approaches. The principle of RPP is to represent the commonalities of different product/part variants on assembly/machining process plans before a product/part variant is specified. The manufacturing process plans for a specific product/part are derived by configuring the existing RPP. In the next chapter, we will explain how to integrate RPP with product/part configuration and a manufacturing system so as to generate the complete and optimal manufacturing process plan for a specific product/part variant.

Chapter 5

Process plan selection and optimization for a specific variant

RPP embodies the commonalities of manufacturing process plans for all the product/part variants on the aspects of process operations and the sequences among these operations. These commonalities can be prepared in the RPP before a specific product/part variant is assigned to a production system. In this way, RPP increases the efficiency of the machining process planning for the product/part variants that are apt to change in a dynamic manufacturing environment. This chapter is devoted to answer the question that how to apply the proposed concepts of RPP under the manufacturing paradigms of MC and co-evolution of P3S. As shown in figure 5-1, two application stages have been identified for RPP to exhibit its capabilities in a dynamic and changeable manufacturing system.

The stage I is the integrated product/part configuration and RPP configuration. In this stage, the modular manufacturing process plans and the corresponding precedence relationships for a specific product/part variant are selected during the configuration process of that product/part variant. They are the building blocks of the final manufacturing process plans for a product/part variant and will be used in the stage II to build the optimal manufacturing process plan.

In the stage II, the process plan elements are reconstructed into the final process plan to meet the optimization objectives in a specific manufacturing scenario. Two manufacturing scenarios are identified for new product/part introduction, and the optimization objectives and the objective functions to evaluate the optimal process plan candidates are defined

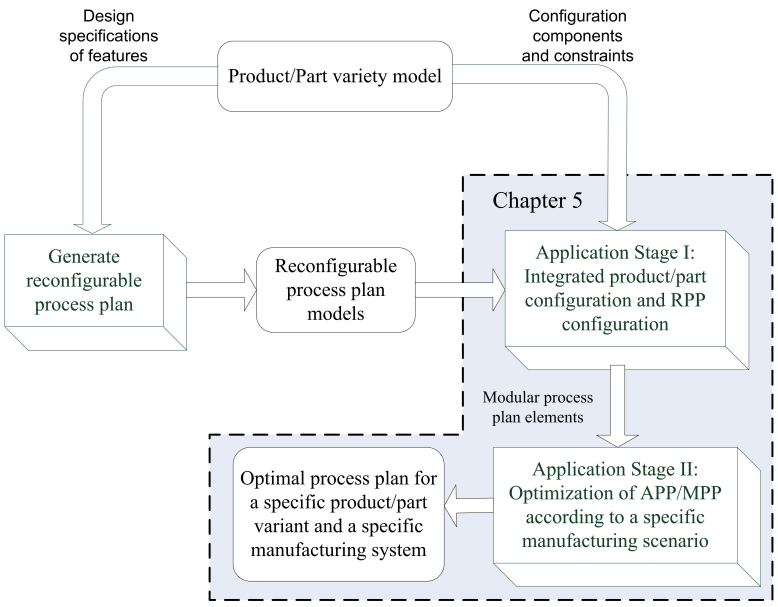


Figure 5-1: Framework of the thesis

respectively for these two scenarios. The optimal manufacturing process plan for current product/part variant is then generated by applying an optimization technique on the solution space of the process planning.

This chapter deals with the two application stages of RPP from two granularity levels: reconfigurable assembly process plan (RAPP) for a product family and reconfigurable machining process plan (RMPP) for a part family. Although the implementations for these two granularity levels are different, the basic principles for their applications are similar.

5.1 Integrated product/part configuration and RPP configuration

Configuration technology has been identified as an enabling technology for MC. Product configuration applies configuration technology on product variant generation, which constrains the set of potential product variants of a product family to the possible combination of predefined components. It has the potential advantages to offer customers the tailored products while ensuring short delivery times (Salvador and Forza, 2004b). Similarly, configuration technology can also be applied on part variant generation which is called part configuration. However, the full potential advantages of product/part configuration can hardly be reaped if there is no responsive process planning to instruct the production system to manufacture

the configured product/part variant.

One application of the proposed RPP is integrating RPP configuration with product/part configuration such that the process plan elements related to a product/part variant can be immediately derived from the RPP when the product/part variant is configured. Because a RPP contains the process plan elements for all the variants in a family, when one variant is derived, only the process plan elements related to this variant should be selected in order to generate the manufacturing process plans for this variant. In terms of RAPP, the process plan elements for a product variant are the modular assembly process plans for the mating modules which have product variety components involved in that product variant; in terms of RMPP, the process plan elements for a part variant include the ROPPs for its feature clusters and the precedence relations between the interactive feature variants on that part variant.

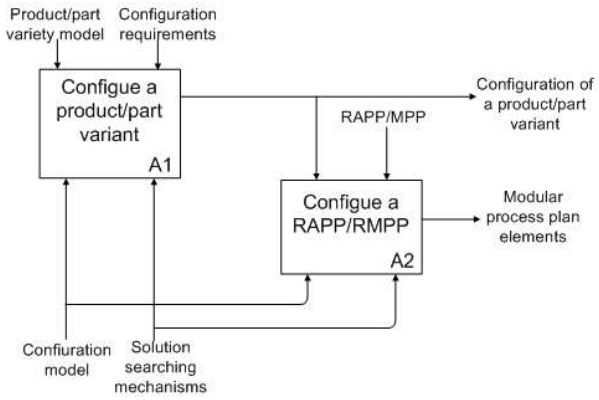


Figure 5-2: IDEF0 diagram for integrated product/part configuration and RPP configuration

As shown in figure 5-2, integrated product/part configuration and RPP configuration takes the configuration requirements for a product/part variant as input, and then it outputs sequentially the configuration of a product/part variant and a set of modular process plan elements for this product/part variant. The configuration of a product/part variant consists of the functional and physical variety components in the product/part variety model satisfying a set of given functional and attributive requirements. In terms of RAPP, the modular process plan elements are a set of modular assembly process plans activated for a product variant; in terms of RMPP, the modular process plan elements refer to a set of RMOPs and precedence relations activated for a part variant.

5.1.1 DCSP-based approach for integrated product/part configuration and RPP configuration

In the early stage of configuration technologies, rule-based configurators were developed to solve product configuration problem. For example, R1/XCON which was developed to support order generation for computer system (McDermott, 1982). At this stage, production rules were used for expressing configuration knowledge. Although rule-based configuration systems had successful application results at that time, they also had two major drawbacks (Felfernig et al., 2014):

- They required enormous efforts in knowledge base development and maintenance;
- The outcome of the configuration strongly depended on the ordering in which different rules were interpreted.

Motivated by the weakness of the rule-based configuration approach, researchers and practitioners turned to develop new configuration technologies over the last decades (Zhang, 2014). These new technological developments can be summarized with the term: model-based approach (Felfernig et al., 2014). The major advantage of model-based approach is a clear separation between domain knowledge and problem solving knowledge. Constraint Satisfaction Problem (CSP) is a widely used model-based knowledge representation formalism. It has the following advantages:

- Simple to use. CSP provides a very simple and convenient way of representing problem and it provides both modeling and solving of a problem within the same framework;
- Standard representation pattern. Assigning values from the domains of variables to variables with the satisfaction of all the constraints is a standard problem representation pattern. This kind of pattern makes CSP domain independent;
- Good efficiency. Constraint propagation technique and backtrack technique make CSP solving process more efficient.

However, classical CSP formalism for configuration problem has a weakness: it has no mechanism to handle the dynamic changes on the set of variants and on the set of constraints during the process of configuration. To improve the weakness of the classical CSP-based approach for configuration, Dynamic Constraint Satisfaction Problem (DCSP)

was introduced (Mittal and Falkenhainer, 1990). The main idea is to only propagate a subset of variables that are relevant to a solution and must be assigned values during the course of problem solving.

In DCSP, every variable can take one of the two states: active and inactive and only active variables are involved into a value assignment process. Beside the states of variables, activity constraints are introduced to specify conditions under which variables become active. The solving process starts with an set of initial active variables and the value assignments on these variables. Other variables are activated into the solving process as soon as the activity constraints involved these variables are satisfied. Correspondingly, a constraint is "active" if all the variables inside this constraint are active; otherwise, it is "inactive". Only active constraints are checked in the problem solving process. With this dynamic mechanism, the searching for the irrelevant variables could be avoided. Integrated product/part configuration and RPP configuration is a dynamic problem in nature. For example, in the hierarchical product architecture in PVDA, if a functional component is not chosen for a variant configuration, then it is not necessary to choose the physical components connecting to this functional component; in addition, for RMOP configuration, if one feature variant is not chosen for a variant configuration, then it is not necessary to choose the precedence relationships involving this feature variant. By considering its dynamic nature, DCSP is adopted to formalize the problem for integrated product/part configuration and RPP configuration. Because RPP has two granularity levels: RAPP and RMPP, we respectively give the representation of DCSP for the problem at each level.

Integrated product configuration and RAPP configuration is a DCSP that is a triple $I =< V, D, C >$ where:

- $V = \{V_{\mathrm{a}}, V_{\mathrm{f}}, V_{\mathrm{sa}}, V_{\mathrm{pf}}, V_{\mathrm{pv}}, V_{\mathrm{ma}}\}$ is a set of variables which is divided into six subsets. V_{a} is a variable set consisting of all the configurable attribute variables, for example, in terms of the gear pump family, they could be variables for model, clearance, relief valve, seal methods. V_{f} is a variable set in which each variable corresponds to a configurable function defined in the product architecture in the PVDA of a product family. V_{sa} is a variable set in which each variable corresponds to a subassembly defined in the product architecture. V_{pf} is a variable set in which each variable corresponds to a configurable part family defined in the product architecture. V_{pv} is a variable set in which each variable corresponds to a configurable part variant defined in the

product architecture. As shown in chapter 2, in the PVDA for a product family, there exist hierarchical relationships between the configurable components at function level, subassembly level, part family level and part variant level. These relationships contribute to compatibility constraints and activity constraints on the variables from V_{a}, V_{f}, V_{sa}, V_{pf} and V_{pv}. V_{ma} is a set of variables in which each variable corresponds to a modular assembly process plan in the RAPP of a product family.

- $D = \{D_{\mathrm{a}}, D_{\mathrm{b}}\}$ is a set of domains for all the variables in V. D_{a} is a set of domains each of which contains all the configurable attribute values for a variable in V_{a}. D_{b} is a boolean domain: $\{0, 1\}$ for all the variables in V_{f},V_{sa},V_{pf},V_{pv} and V_{ma}. If the variables in V_{f}, V_{sa}, V_{pf}, V_{pv} and V_{ma} take the value of 0 from D_{b}, then the configurable components corresponding to these variables are not chosen in the final configuration, otherwise, the variables take the value of 1, which means the configurable components are chosen in the final configuration.

- $C = \{C_{\mathrm{c}}, C_{\mathrm{a}}\}$ is a set of constraints each of which constrains the values of the variables in V. C is divided into two subsets, C_{c} and C_{a}. C_{c} is a set of compatibility constraints on the variables in V and C_{a} is a set of activity constraints. The compatibility constraints in C_{c} define the selectivity relations among the configurable components of a product family. The activity constraints in C_{a} describe conditions under which a variable may or may not be actively considered as a part of a final solution. Both compatibility constraints and activity constraints can be expressed by the propositional-logic-based representation scheme proposed in chapter 2.

Similarly, Integrated part configuration and RMPP configuration is also a DCSP that is a triple $I =< V, D, C >$ where (Xia et al., 2016):

- V=$\{V_{\mathrm{f}}$, V_{fc}, V_{fv}, V_{rm}, $V_{\mathrm{pr}}\}$ is a set of variables. It consists of four subsets, V_{f}, V_{fc}, V_{fv}, V_{rm} and V_{pr}. V_{f} is a set of variables corresponding to the function modules in the part variety decomposition network. V_{fc} is a set of variables corresponding to the feature clusters in the part variety decomposition network. V_{fv} is a set of variables corresponding to the feature variants in the part variety decomposition network. V_{rm} consists of a set of variables corresponding to the RMOPs in the RMPP. V_{pr} includes a set of variables corresponding to the precedence relations among the physical variety components of a part family. All the variables in V are Boolean variables.

- $D = \{0, 1\}$ is a Boolean domain for all the variables in V.

- Like the DCSP for integrated product configuration and RAPP configuration, $C = \{C_c, C_a\}$ is a set of constraints where C_c is a set of compatibility constraints and C_a is a set of activity constraints.

5.1.2 Example for integrated product configuration and RAPP configuration

To illustrate the DCSP formalism for integrated product configuration and RAPP configuration, the gear pump family mentioned in chapter 2 is used as an example. As shown in figure 2-4, the set of the configurable attributes for this example is {model, clearance, relief valve, seal method, lubrication methods and port types}. In order to simplify our example without losing the generality, we assume that there are three gear pump variants in this family and their attributes are listed in table 5.1.

Table 5.1: Attributes of three gear pump variants in a gear pump body family

Product variants	Model (v_m)	Clearance (v_c)	Relief valve (v_{rv})	Seal method (v_{sm})	Lubrication method (v_{lm})	Port types (v_{pt})
Variant 1	G1-2	A	No	Packing	Grease	NPT standard
Variant 2	G1-4	B	Yes	Mechanical seal	Oil	NPT standard
Variant 3	G1-55	C	Yes	Packing	Oil	ANSI standard

A = 19.0 mm; B = 44.4 mm; C = 63.5 mm.

According to the attributes of the three product variants, they have different configurations on their parts. The parts of these three product variants are listed in the table B.1. Variant 1 and Variant 2 share a part of common parts, while all the parts of variant 3 are different with those of variant 1 and variant 2 except grease fitting 1, which is shared by all the variants.

Figure 5-3 shows the functional structure and the mapping between functional components and physical components of the gear pump family. To ease this example, only functions and subassemblies are shown in figure 5-3. The mapping relations for the part families and part variants of this gear pump family are shown separately in the figure A-1-A-8 in Ap-

pendix B.

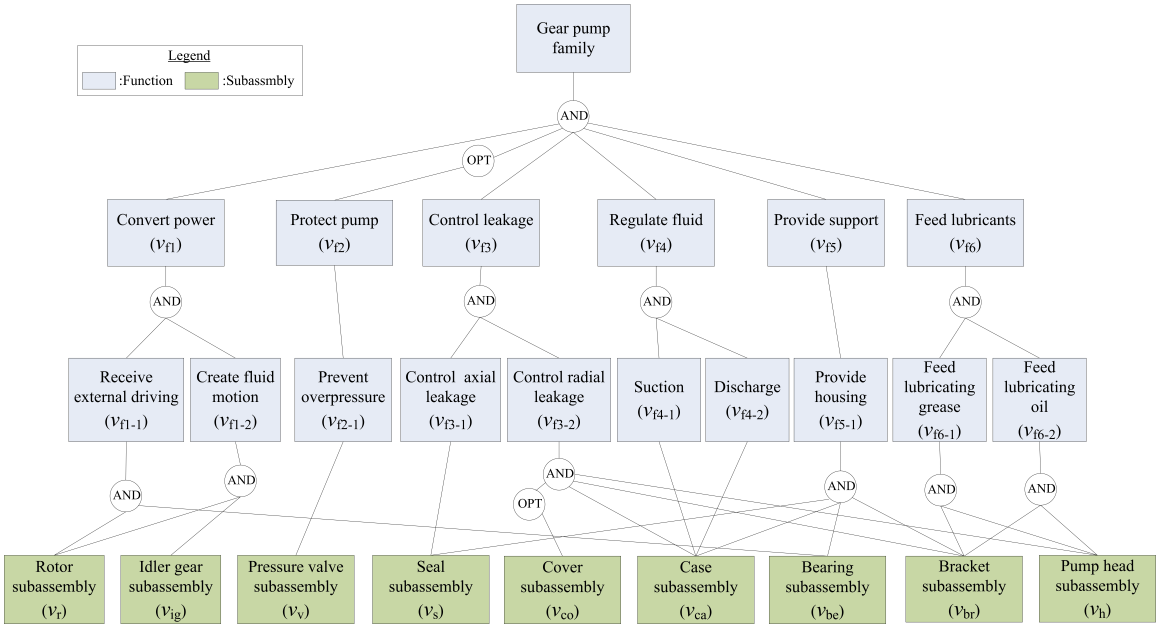


Figure 5-3: Product architecture defined in the PVDA of the gear pump family (part family level and part variant level are not shown)

According to chapter 3, the RAPP for this pump body family consists of the modular assembly process plans for the mating modules whose mating relations are represented by the connected DFC models. There are totally 11 modular assembly process plans in the RAPP for this pump body family, two of them are represented at product family level and nine of them are represented at product variant level. Each modular assembly process plan is represented by a AND/OR graph. In the DCSP, each modular assembly process plan is assigned with one variable, thus, for this example, we assign: v_{ma1} to G^1_{PF} which is the AND/OR graph shown in figure 3-1, v_{ma2} to G^2_{PF} which is the AND/OR graph shown in figure 3-2, and v_{ma3}, v_{ma4}, v_{ma5}, v_{ma6}, v_{ma7}, v_{ma8}, v_{ma9}, v_{ma10}, v_{ma11} respectively to G^1_{PV}, G^2_{PV},G^3_{PV}, G^4_{PV}, G^5_{PV}, G^6_{PV}, G^7_{PV}, G^8_{PV} and G^9_{PV}.

Based on the above descriptions, we can formalize the set of variables V=$\{V_{\mathrm{a}}, V_{\mathrm{f}}, V_{\mathrm{sa}}, V_{\mathrm{pf}}, V_{\mathrm{pv}}, V_{\mathrm{ma}}\}$ in the DCSP for integrated product and RAPP configuration as follows:

- $V_{\mathrm{a}} = \{v_{\mathrm{m}}, v_{\mathrm{c}}, v_{\mathrm{rv}}, v_{\mathrm{sm}}, v_{\mathrm{lb}}, v_{\mathrm{pt}}\}$, corresponding to the six attributes in table 5-1;
- V_{f}= $\{v_{\mathrm{f1}}, v_{\mathrm{f2}}, v_{\mathrm{f3}}, v_{\mathrm{f4}}, v_{\mathrm{f5}}, v_{\mathrm{f6}}, v_{\mathrm{f1\text{-}1}}, v_{\mathrm{f1\text{-}2}}, v_{\mathrm{f2\text{-}1}}, v_{\mathrm{f3\text{-}1}}, v_{\mathrm{f3\text{-}2}}, v_{\mathrm{f4\text{-}1}}, v_{\mathrm{f4\text{-}2}}, v_{\mathrm{f5\text{-}1}}, v_{\mathrm{f6\text{-}1}}, v_{\mathrm{f6\text{-}2}}\}$, corresponding to the functions shown in figure 5-3;

- V_{sa}= $\{v_{\mathrm{be}}, v_{\mathrm{br}}, v_{\mathrm{ca}}, v_{\mathrm{r}}, v_{\mathrm{ig}}, v_{\mathrm{h}}, v_{\mathrm{s}}, v_{\mathrm{v}}, v_{\mathrm{co}}\}$, corresponding to the subassemblies listed in table B.1;

- V_{pf}=$\{v_{\mathrm{be1}}, v_{\mathrm{be2}}, v_{\mathrm{be3}}, v_{\mathrm{be4}}, v_{\mathrm{be5}}, v_{\mathrm{be6}}, v_{\mathrm{be7}}, v_{\mathrm{be8}}, v_{\mathrm{be9}}, v_{\mathrm{br1}}, v_{\mathrm{br2}}, v_{\mathrm{br3}}, v_{\mathrm{br4}}, v_{\mathrm{br5}}, v_{\mathrm{br6}}, v_{\mathrm{br7}}, v_{\mathrm{br8}}, v_{\mathrm{ca1}}, v_{\mathrm{ca2}}, v_{\mathrm{r1}}, v_{\mathrm{r2}}, v_{\mathrm{ig1}}, v_{\mathrm{ig2}}, v_{\mathrm{h1}}, v_{\mathrm{h2}}, v_{\mathrm{h3}}, v_{\mathrm{h4}}, v_{\mathrm{h5}}, v_{\mathrm{h6}}, v_{\mathrm{s1}}, v_{\mathrm{s2}}, v_{\mathrm{s3}}, v_{\mathrm{s4}}, v_{\mathrm{s5}}, v_{\mathrm{s6}}, v_{\mathrm{s7}}, v_{\mathrm{v1}}, v_{\mathrm{v2}}, v_{\mathrm{v3}}, v_{\mathrm{v4}}, v_{\mathrm{v5}}, v_{\mathrm{v6}}, v_{\mathrm{v7}}, v_{\mathrm{v8}}, v_{\mathrm{v9}}, v_{\mathrm{v10}}, v_{\mathrm{v11}}, v_{\mathrm{co1}}\}$, corresponding to the part families listed in table B.1;

- V_{pv}=$\{v_{\mathrm{be1\text{-}1}}, v_{\mathrm{be1\text{-}2}}, v_{\mathrm{be2\text{-}1}}, v_{\mathrm{be2\text{-}2}}, v_{\mathrm{be3\text{-}1}}, v_{\mathrm{be3\text{-}2}}, v_{\mathrm{be4\text{-}1}}, v_{\mathrm{be4\text{-}2}}, v_{\mathrm{be5\text{-}1}}, v_{\mathrm{be5\text{-}2}}, v_{\mathrm{be6\text{-}1}}, v_{\mathrm{be6\text{-}2}}, v_{\mathrm{be7\text{-}1}}, v_{\mathrm{be7\text{-}2}}, v_{\mathrm{be8\text{-}1}}, v_{\mathrm{be8\text{-}2}}, v_{\mathrm{be9\text{-}1}}, v_{\mathrm{be9\text{-}2}}, v_{\mathrm{br1\text{-}1}}, v_{\mathrm{br1\text{-}2}}, v_{\mathrm{br2\text{-}1}}, v_{\mathrm{br2\text{-}2}}, v_{\mathrm{br2\text{-}3}}, v_{\mathrm{br3\text{-}1}}, v_{\mathrm{br4\text{-}1}}, v_{\mathrm{br4\text{-}2}}, v_{\mathrm{br6\text{-}1}}, v_{\mathrm{br7\text{-}1}}, v_{\mathrm{br7\text{-}2}}, v_{\mathrm{br8\text{-}1}}, v_{\mathrm{ca1\text{-}1}}, v_{\mathrm{br1\text{-}2}}, v_{\mathrm{ca2\text{-}1}}, v_{\mathrm{ca2\text{-}2}}, v_{\mathrm{r1\text{-}1}}, v_{\mathrm{r1\text{-}2}}, v_{\mathrm{r1\text{-}3}}, v_{\mathrm{r2\text{-}1}}, v_{\mathrm{ig1\text{-}1}}, v_{\mathrm{ig1\text{-}2}}, v_{\mathrm{ig1\text{-}3}}, v_{\mathrm{ig2\text{-}1}}, v_{\mathrm{ig2\text{-}2}}, v_{\mathrm{ig2\text{-}3}}, v_{\mathrm{h1\text{-}1}}, v_{\mathrm{h1\text{-}2}}, v_{\mathrm{h1\text{-}3}}, v_{\mathrm{h2\text{-}1}}, v_{\mathrm{h2\text{-}2}}, v_{\mathrm{h2\text{-}3}}, v_{\mathrm{h3\text{-}1}}, v_{\mathrm{h3\text{-}2}}, v_{\mathrm{h4\text{-}1}}, v_{\mathrm{h4\text{-}2}}, v_{\mathrm{h5\text{-}1}}, v_{\mathrm{h6\text{-}1}}, v_{\mathrm{s1\text{-}1}}, v_{\mathrm{s1\text{-}2}}, v_{\mathrm{s2\text{-}1}}, v_{\mathrm{s2\text{-}2}}, v_{\mathrm{s3\text{-}1}}, v_{\mathrm{s3\text{-}2}}, v_{\mathrm{s4\text{-}1}}, v_{\mathrm{s4\text{-}2}}, v_{\mathrm{s5\text{-}1}}, v_{\mathrm{s6\text{-}1}}, v_{\mathrm{s6\text{-}2}}, v_{\mathrm{s7\text{-}1}}, v_{\mathrm{v1\text{-}1}}, v_{\mathrm{v1\text{-}2}}, v_{\mathrm{v2\text{-}1}}, v_{\mathrm{v2\text{-}2}}, v_{\mathrm{v3\text{-}1}}, v_{\mathrm{v3\text{-}2}}, v_{\mathrm{v4\text{-}1}}, v_{\mathrm{v4\text{-}2}}, v_{\mathrm{v5\text{-}1}}, v_{\mathrm{v5\text{-}2}}, v_{\mathrm{v6\text{-}1}}, v_{\mathrm{v6\text{-}2}}, v_{\mathrm{v7\text{-}1}}, v_{\mathrm{v7\text{-}2}}, v_{\mathrm{v8\text{-}1}}, v_{\mathrm{v8\text{-}2}}, v_{\mathrm{v9\text{-}1}}, v_{\mathrm{v9\text{-}2}}, v_{\mathrm{v10\text{-}1}}, v_{\mathrm{v10\text{-}2}}, v_{\mathrm{v11\text{-}1}}, v_{\mathrm{v11\text{-}2}}, v_{\mathrm{v11\text{-}3}}, v_{\mathrm{co1}}\}$, corresponding to the part variants listed in table B.1;

- V_{ma}= $\{v_{\mathrm{ma1}}, v_{\mathrm{ma2}}, v_{\mathrm{ma3}}, v_{\mathrm{ma4}}, v_{\mathrm{ma5}}, v_{\mathrm{ma6}}, v_{\mathrm{ma7}}, v_{\mathrm{ma8}}, v_{\mathrm{ma9}}, v_{\mathrm{ma10}}, v_{\mathrm{ma11}}\}$, corresponding to the modular assembly process plans in figure 3-1, 3-2 and 3-3.

Then, the set of domains $D = \{D_{\mathrm{a}}, D_{\mathrm{b}}\}$ for the variables in V in the DCSP for integrated product and RAPP configuration is formalized as follows:

- D_{a}=$\{D_{\mathrm{m}}, D_{\mathrm{c}}, D_{\mathrm{rv\text{-}lb}}, D_{\mathrm{sm}}, D_{\mathrm{pt}}\}$, in which D_{m}= {"G1−2", "G1−4", G1−55 }, D_{c}= {"A", "B", "C"}, $D_{\mathrm{rv\text{-}lb}}$={"Grease", "Oil"}, D_{sm}={"Packing", "Mechanical seal"}, D_{pt}={"NPT standard", "ANSI standard"}. As listed in table 5.1, D_{m} is the domain for the attribute variable v_{m}; D_{c} is the domain for the attribute variable v_{c}; $D_{\mathrm{rv\text{-}lb}}$ is the domain for the attribute variable v_{rv} and v_{lb}; D_{sm} is the domain for the attribute variable v_{sm}; and D_{pt} is the domain for the attribute variable v_{pt};

- D_b=$\{0, 1\}$ and $\forall v \in V_{\mathrm{f}}, V_{\mathrm{sa}}, V_{\mathrm{pf}}, V_{\mathrm{pv}}, V_{\mathrm{ma}}$($D_{\mathrm{b}}$ is a domain for v).

The compatibility constraints in C_{c} on the variables in V=$\{V_{\mathrm{a}}, V_{\mathrm{f}}, V_{\mathrm{sa}}, V_{\mathrm{pf}}, V_{\mathrm{pv}}, V_{\mathrm{ma}}\}$ come from five aspects :

1. The configuration relations on the child components of the same parent component from the upper decomposition level in the PVDA of the gear pump family. These

configuration relations are represented by the three types of logical operators. The variables involved in these configuration relations are the variables in V_{f}, V_{sa}, V_{pf}, and V_{pv}. For example, in figure 5-3, an **AND** operator links two subfunctions: $V_{\mathrm{f1\text{-}1}}$ and $V_{\mathrm{f1\text{-}2}}$ to one upper level function: V_{f1}, thus, this logical operator contributes to one compatibility constraint:

$$v_{\mathrm{f1}} \leftrightarrow v_{\mathrm{f1\text{-}1}} \wedge v_{\mathrm{f1\text{-}2}}.$$

2. The configuration relations on the child components of different parent components from the upper decomposition level. The variables involved in these configuration relations are also the variables in V_{f}, V_{sa}, V_{pf}, and V_{pv}. For example, at the subassembly level, one gear variant must choose either pressure relief valve subassembly and cover subassembly, it cannot have both of them:

$$v_{\mathrm{v}} \leftrightarrow \neg v_{\mathrm{co}}.$$

3. The configuration relations on the attributes of the gear pump. The variables involved in these configuration relations are the variables in V_{a}. For example, the attribute configuration of first gear pump variant in table 5.1 forms the following compatibility constraint:

$$\begin{aligned} v_{\mathrm{m}} = \text{"G1-2"} \quad \leftrightarrow \quad & v_{\mathrm{c}} = \text{"A"} \wedge v_{\mathrm{rv}} = \text{"No"} \wedge v_{\mathrm{sm}} = \text{"Packing"} \wedge v_{\mathrm{lb}} = \text{"Grease"} \wedge \\ & v_{\mathrm{pt}} = \text{"NPT standard"}. \end{aligned}$$

4. The configuration relations between the attributes and the configurable components in the PVDA of the gear pump family. The variables involved in these configuration relations could be the variables in V_{a}, V_{f}, V_{sa}, V_{pf}, and V_{pv}. For example, for a gear pump variant which uses grease as the lubrication method, grease fitting 2 and grease fitting 3 should be chosen:

$$v_{\mathrm{lb}} = \text{"Grease"} \leftrightarrow v_{\mathrm{br8}} \wedge v_{\mathrm{h6}}.$$

5. The configuration relations between the physical components of the gear pump family and the modular assembly process plans in the RAPP for the mating modules of the

gear pump family. The variables involved in these configuration relations could be the variables in V_{sa}, V_{pf}, V_{pv}, and V_{ma}, for example, from figure 3-1 there are the following compatibility constraint on v_{be},v_{br}, v_{ca}, v_{r}, v_{ig}, v_{h}, v_{s}, v_{v10}, v_{v11} and v_{ma1}:

$$v_{\mathrm{be}} \wedge v_{\mathrm{br}} \wedge v_{\mathrm{ca}} \wedge v_{\mathrm{r}} \wedge v_{\mathrm{ig}} \wedge v_{\mathrm{h}} \wedge v_{\mathrm{s}} \wedge v_{\mathrm{v10}} \wedge v_{\mathrm{v11}} \leftrightarrow v_{\mathrm{ma1}}.$$

The activity constraints in C_{a} define that the conditions in which the variables are activated. Only the compatibility constraints in which all the variables are activated can be explored by the solution searching process.

In the DCSP for integrated product configuration and RAPP configuration, there are a set of initial variables which takes the initial inputs at the beginning of the problem solving process. In this gear pump body example, the initial variables include the attribute variables V_{a}=$\{v_{\mathrm{m}}, v_{\mathrm{c}}, v_{\mathrm{rv}}, v_{\mathrm{sm}}, v_{\mathrm{lb}}, v_{\mathrm{pt}}\}$ and the variables in $\{v_{\mathrm{f1}}, v_{\mathrm{f2}}, v_{\mathrm{f3}}, v_{\mathrm{f4}}, v_{\mathrm{f5}}, v_{\mathrm{f6}}\}$, which correspond to the functions at the top level of the function structure in the PVDA.

The activity constraints on the variables in V_{f}, V_{sa}, V_{pf} and V_{pv} are formulated according to hierarchical relationships between the configurable components in the PVDA of the gear pump family. In the PVDA, the configurable components at the upper level determine the activation of the components at the lower level connecting to these upper-level components. Only when the variable of an upper-level component takes the value of "1", its lower-level components are activated and the variable of these lower-level components can then be assigned values. Otherwise, its lower-level components are inactivated and the constraints involving them will not be explored by the solution searching process. For example, from figure A-3, the following activity constraints can be derived:

$$\begin{aligned} v_{\mathrm{be}} = 1 \quad \leftrightarrow \quad & \mathrm{Active}{:}v_{\mathrm{be1}} \wedge \mathrm{Active}{:}v_{\mathrm{be2}} \wedge \mathrm{Active}{:}v_{\mathrm{be3}} \wedge \mathrm{Active}{:}v_{\mathrm{be4}} \wedge \\ & \mathrm{Active}{:}v_{\mathrm{be5}} \wedge \mathrm{Active}{:}v_{\mathrm{be6}} \wedge \mathrm{Active}{:}v_{\mathrm{be7}} \wedge \mathrm{Active}{:}v_{\mathrm{be8}} \wedge \mathrm{Active}{:}v_{\mathrm{be9}}. \end{aligned}$$

5.1.3 Example for integrated part configuration and RMPP configuration

To illustrate the DCSP formalism for integrated part configuration and RMPP configuration, we use the oil pump body family mentioned in chapter 2 as an example. As shown in figure 2-7, there are two part variants in this part family. The variety components are structured in the part variety decomposition network of this part family shown in figure 2-8. There are three decomposition levels in the part variety decomposition network, function module

level, feature cluster level and feature variants level. Each component in the part variety decomposition network corresponds to a variable in the DCSP. The RMOPs and precedence relations in the RMPP of this gear pump family mentioned in chapter 3 also contribute to the variables in the DCSP.

The compatibility constraints of the DCSP come from two aspects. The first aspect is the configuration constraints defined in the part variety model of this part family; the second one is the mapping relations between the physical components of the part family and the components of the RMOP of the part family. The hierarchical relations between the variety components in the decomposition network contribute to the activity constraints of the DCSP.

Based on the above descriptions, we can formalize the DCSP $I =< V, D, C >$ for this oil pump body family. The variables in V are listed in the table in Appendix B.2 with the corresponding variety components of the part family. The domain for these variables is a Boolean domain $\{0, 1\}$. Only the domain of an activated variable will be explored by the solution searching process.

In this DCSP, the variables in $V_{\mathrm{f}} = \{v_{\mathrm{f1}}, v_{\mathrm{f2}}, v_{\mathrm{f3}}, v_{\mathrm{f4}}\}$ are a set of initial variables. The DCSP solving process takes the values of these initial variables and then find the configuration which satisfies all the constraints involved by the values of these initial variables.

After defining the variables for the variety components in this DCSP, the compatibility constraints on these variables are formulated according to their configuration relations. For the compatibility constraints come from the configuration constraints defined in the part variety model of this oil pump body family, there are the following compatibility constraints:

C_{c1}: $1 \equiv (v_{\mathrm{f1}} \wedge v_{\mathrm{f2}} \wedge v_{\mathrm{f3}} \wedge \neg v_{\mathrm{f4}}) \vee (v_{\mathrm{f1}} \wedge v_{\mathrm{f2}} \wedge \neg v_{\mathrm{f3}} \wedge v_{\mathrm{f4}})$;
C_{c2}: $v_{\mathrm{f1}} \leftrightarrow v_{\mathrm{pc1}} \wedge v_{\mathrm{cc1}}$;
C_{c3}: $v_{\mathrm{f2}} \leftrightarrow (v_{\mathrm{pc2}} \wedge v_{\mathrm{hc2}}) \vee (\neg v_{\mathrm{pc2}} \wedge v_{\mathrm{hc2}})$;
C_{c4}: $v_{\mathrm{f3}} \leftrightarrow v_{\mathrm{hc3}}$;
C_{c5}: $v_{\mathrm{f2}} \leftrightarrow v_{\mathrm{pc4}} \wedge v_{\mathrm{cc4}}$;
C_{c6}: $v_{\mathrm{pc1}} \leftrightarrow (v_{\mathrm{pc1\text{-}1}} \wedge \neg v_{\mathrm{pc1\text{-}2}}) \vee (\neg v_{\mathrm{pc1\text{-}1}} \wedge v_{\mathrm{pc1\text{-}2}})$;
C_{c7}: $v_{\mathrm{cc1}} \leftrightarrow (v_{\mathrm{cc1\text{-}1}} \wedge \neg v_{\mathrm{cc1\text{-}2}}) \vee (\neg v_{\mathrm{cc1\text{-}1}} \wedge v_{\mathrm{cc1\text{-}2}})$;
C_{c8}: $v_{\mathrm{pc2}} \leftrightarrow v_{\mathrm{pc2\text{-}1}}$;
C_{c9}: $v_{\mathrm{hc2}} \leftrightarrow (v_{\mathrm{hc1\text{-}1}} \wedge \neg v_{\mathrm{hc1\text{-}2}}) \vee (\neg v_{\mathrm{hc1\text{-}1}} \wedge v_{\mathrm{hc1\text{-}2}})$;
C_{c10}: $v_{\mathrm{hc3}} \leftrightarrow v_{\mathrm{hc3\text{-}1}}$;
C_{c11}: $v_{\mathrm{pc4}} \leftrightarrow (v_{\mathrm{pc4\text{-}1}} \wedge v_{\mathrm{pc4\text{-}2}})$;
C_{c12}: $v_{\mathrm{cc4}} \leftrightarrow v_{\mathrm{cc4\text{-}1}}$;
C_{c13}: $v_{\mathrm{pc1\text{-}1}} \leftrightarrow v_{\mathrm{cc1\text{-}1}}$;
C_{c14}: $v_{\mathrm{pc1\text{-}2}} \leftrightarrow v_{\mathrm{pc2\text{-}1}}$;

C_{c15}: $v_{pc1\text{-}1} \leftrightarrow v_{hc3\text{-}1}$;
C_{c16}: $v_{pc1\text{-}1} \leftrightarrow v_{hc2\text{-}1}$;
C_{c17}: $v_{pc1\text{-}2} \leftrightarrow v_{f4}$;
C_{c18}: $v_{pc1\text{-}2} \leftrightarrow v_{pc2}$.

For the compatibility constraints come from the mapping relations between the physical components of the part family and the components of the RMOP of the part family, there are the following compatibility constraints :

C_{c19}: $v_{pc1} \leftrightarrow v_{rm1}$;
C_{c20}: $v_{cc1} \leftrightarrow v_{rm2}$;
C_{c21}: $v_{pc2} \leftrightarrow v_{rm3}$;
C_{c22}: $v_{hc2} \leftrightarrow v_{rm4}$;
C_{c23}: $v_{hc3} \leftrightarrow v_{rm5}$;
C_{c24}: $v_{pc4} \leftrightarrow v_{rm6}$;
C_{c25}: $v_{cc4} \leftrightarrow v_{rm7}$;
C_{c26}: $v_{hc2} \wedge v_{pc1} \leftrightarrow v_{pr1}$;
C_{c27}: $v_{pc1} \wedge v_{cc1} \leftrightarrow v_{pr2}$;
C_{c28}: $v_{hc2} \wedge v_{hc3} \leftrightarrow v_{pr3}$;
C_{c29}: $v_{hc2} \wedge v_{pc4} \leftrightarrow v_{pr4}$;
C_{c30}: $v_{pc4} \wedge v_{cc4} \leftrightarrow v_{pr5}$;
C_{c31}: $v_{pc4} \leftrightarrow v_{pr6}$.

According to the hierarchical relations between the configurable components in the decomposition network, the following activity constraints in this DCSP are formulated:

C_{a1}: $v_{f1} = 1 \leftrightarrow \text{Active:}v_{pc1} \wedge \text{Active:}v_{cc1}$;
C_{a2}: $v_{f2} = 1 \leftrightarrow \text{Active:}v_{pc2} \wedge \text{Active:}v_{cc2}$;
C_{a3}: $v_{f3} = 1 \leftrightarrow \text{Active:}v_{hc3}$;
C_{a4}: $v_{f4} = 1 \leftrightarrow \text{Active:}v_{pc4} \wedge \text{Active:}v_{cc4}$;
C_{a5}: $v_{pc1} = 1 \leftrightarrow \text{Active:}v_{pc1\text{-}1} \wedge \text{Active:}v_{pc1\text{-}2}$;
C_{a6}: $v_{cc1} = 1 \leftrightarrow \text{Active:}v_{cc1\text{-}1} \wedge \text{Active:}v_{cc1\text{-}2}$;
C_{a7}: $v_{pc2} = 1 \leftrightarrow \text{Active:}v_{pc2\text{-}1}$;
C_{a8}: $v_{hc2} = 1 \leftrightarrow \text{Active:}v_{cc2\text{-}1} \wedge \text{Active:}v_{cc2\text{-}2}$;
C_{a9}: $v_{hc3} = 1 \leftrightarrow \text{Active:}v_{hc3\text{-}1}$;
C_{a10}: $v_{pc4} = 1 \leftrightarrow \text{Active:}v_{pc4\text{-}1} \wedge \text{Active:}v_{pc4\text{-}2}$;
C_{a11}: $v_{cc4} = 1 \leftrightarrow \text{Active:}v_{cc4\text{-}1}$.

5.1.4 Implementation

The proposed DCSP-based approach for integrated product/part configuration and RPP configuration is implemented by using a constraint logic programming system named ECLiPSe. ECLiPSe is an open-source software system for development and deployment of constraint

programming applications in the areas of planning, scheduling and configuration. It contains several constraint solver libraries. By using these libraries, the system applies constraint propagation techniques to find the values for the variables in a CSP. For this implementation, we use a finite domain solver named ic library. The programming language used in ECLiPSe is a Prolog-based constraint logic programming language and some useful predicates are specially defined in this system. In addition, ECLiPSe has an integrated development environment named TkECLiPSe. The user interface of TkECLiPSe is shown in figure 5-4.

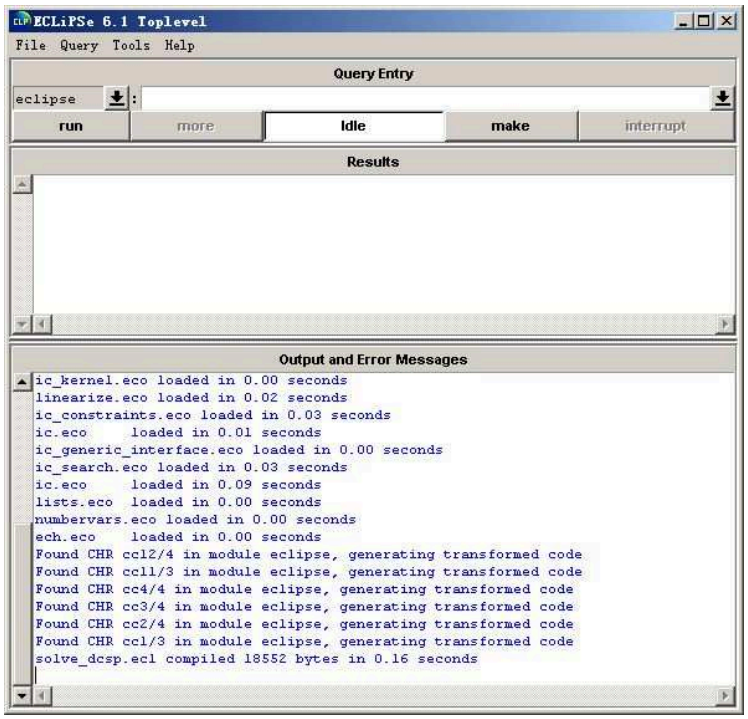


Figure 5-4: User interface of TkECLiPSe

Integrated product/part configuration and RPP configuration is implemented as a constraint program in ECLiPSe system. The typical top-level structure of a constraint program in the system is:

```
Head:solve_problem(Inputs,Outputs):- %A predicate used to invoke the problem
    solving process
   Body part 1 %Specify the variables and the domains of these variables
   Body part 2 %Specify the constraints of the problem
   Body part 3 %Specify the method used for searching the solution
Tail %Define the problem-specific predicates used in the program body
```

The head of the program is a predicate used to invoke the problem solving process. It passes the inputs of problem to the CSP solver and also delivers the solutions from the solver. The body of the program consists of three parts: the first part is for the specifications of variables and their domains; the second part is for the specifications of all the constraints; and the third part is for the specifications of the way used for searching the solution. For a simple problem, an enumerative searching method, like using the predicate - *labeling*, is enough, while for a large problem, an incomplete search maybe needed, such as simulated annealing or Tabu search. The tail of the program consists of the predicate definitions of the problem-specific predicates used in the body; these definitions indicate the executions of the problem solving process when it encounters these problem-specific predicates.

As an illustration, the implementation for integrated part configuration and RMPP configuration is shown here. The implementation for integrated product configuration and RAPP configuration has the same steps but with more variables and constraints.

For the implementation, the following structures are defined to represent the objects of our problem in the program:

- For the oil pump body family:
  :- *local struct(part_family(fms, fcs, fvs))*;

- For the RMPP for the oil pump body family:
  :- *local struct(rmpp(rms, prs))*;

- For the function modules at function module level: :- *local struct(fms(f1, f2, f3, f4))*;

- For the feature clusters at feature cluster level: :- *local struct(fcs(pc1, cc1, hc2, pc2, hc3, pc4, cc4))*;

- For the feature variants at feature variant level: :- *local struct(fvs(pc1_1, pc1_2, cc1_1, cc1_2, hc2_1, hc2_2, pc2_1, hc3_1, pc4_1, pc4_2, cc4_1))*;

- For the RMOPs in the RMPP: :- *local struct(rms(rm1, rm2, rm3, rm4, rm5, rm6, rm7))*;

- For the precedence relations in the RMPP: :- *local struct(prs(pr1, pr2, pr3, pr4, pr5, pr6))*.

In ECLiPSe, the logical variables are written with an upper-case letter or an underscore at the beginning. According to the problem model for integrated part configuration and RMPP configuration in section 5.1.3, we define the variables in the system as follows and each variable corresponds to a configurable component in the problem:

- For the function modules: *f1:F1, f2:F2, f3:F3, f4:F4*;
- For the feature clusters: *pc1:PC1, cc1:CC1, hc2:HC2, pc2:PC2, hc3:HC3, pc4:PC4, cc4:CC4*;
- For the feature variants: *pc1_1:PC1_1, pc1_2:PC1_2, cc1_1:CC1_1, cc1_2:CC1_2, hc2_1:HC2_1, hc2_2:HC2_2, pc2_1:PC2_1, hc3_1:HC3_1, pc4_1:PC4_1, pc4_2:PC4_2, cc4_1:CC4_1*;
- For the RMOPs: *rm1:RM1, rm2:RM2, rm3:RM3, rm4:RM4, rm5:RM5, rm6:RM6, rm7:RM7*;
- For the precedence relations: *pr1:PR1, pr2:PR2, pr3:PR3, pr4:PR4, pr5:PR5, pr6:PR6.*

Since all the variables in this problem are Boolean variables, a Boolean domain is assigned for all these variables:

- *[F1,F2,F3,F4]::[0,1]*;
- *[PC1,CC1,HC2,PC2,HC3,PC4,CC4]::[0,1]*;
- *[PC1_1,PC1_2,CC1_1,CC1_2,HC2_1,HC2_2,PC2_1,HC3_1,PC4_1,PC4_2,CC4_1]::[0,1]*;
- *[RM1,RM2,RM3,RM4,RM5,RM6,RM7]::[0,1]*,
- *[PR1,PR2,PR3,PR4,PR5,PR6]::[0,1].*

A predicate named *activate/4* is defined to dynamically feedback a list of currently active variables to the solution searching process. The list is used by the process to judge whether or not a constraint is active. The activity constraints are implemented by using this predicate. The code for defining this predicate in the program is listed as follow:

```
activate(_:H, L2, Active, New_active)}:-
   ((nonvar(H),H =1) -> add(L2,Active,New_active); New_active = Active)
```

that means if H is not a variable and H =1, then put $L2$ into a list *Active* to form a new list *New_ active*; else the new list *New_ active* equals the old *Active*.

In the DCSP, a compatibility constraint is activated when all the variables in this constraint are active. In order to implement this activation mechanism for the compatibility constraints, a declarative high-level constraint logic program language - Constraint Handling Rule is used for defining the compatibility constraints in the DCSP. By using this language, one compatibility constraint can be delayed if one of its variables is not active.

Constraint handling rule defines two main kinds of constraint handling rules, Simplification and Propagation. Simplification replaces constraints by simpler constraints while preserving logical equivalence. Propagation adds new constrains which are logically redundant but may cause further simplification. In our implementation, we use a series of simplification rules to define the activation mechanism for the compatibility constraints. The simplification rules have the formulation of : *Head <=> Guard | Body.* In our implementation, the head is a representation defined for a class of compatibility constraints which have the same constraint structure (the same number of variables and the same logical relation on the variables); the activation conditions of a compatibility constraint are expressed in Guard, while the body is a logical clause for the class of compatibility constraints represented by the head. When a compatibility constraint is posted as an instance of the head, the guard is checked to determine whether the rule is fired. Once the rule is fired, the head is replaced by the body and then the logical clause is activated and acts as a compatibility constraint in the problem solving process.

According to the compatibility constraints that are modeled in section 5.1.3, we define the following constraint handling rules in the program, where *memberd/2* is a predicate to test whether a variable is active or not:

- *cc1(A:L, B:M, Active)<=>memberd(A:L, Active), memberd(B:M, Active)|L $= M*;
- *cc2(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active), memberd(C:N, Active)|L $= (M and N)*;
- *cc3(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active), memberd(C:N, Active)|L $= ((M and N) or (neg M and N))*;
- *cc4(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active), memberd(C:N, Active)|L $= ((M and neg N) or (neg M and N))*;

- *cc11(A:L, B:M, Active)<=>memberd(A:L, Active)/L $= M*;
- *cc11(A:L, B:M, Active)<=>not memberd(A:L, Active)/M = 0*;
- *cc12(A:L,B:M,C:N,Active)<=>memberd(A:L,Active),memberd(B:M,Active)/N $= (L and M)*;
- *cc12(A:L, B:M, C:N, Active)<=>not memberd(A:L, Active)/N = 0*;
- *cc12(A:L, B:M, C:N, Active)<=>not memberd(B:M, Active)/N = 0.*

After defining these constraint handling rules, any compatibility constraint in our problem is stated as an instance of the head of the corresponding rule. For example, $C_{c2}(v_{f1} \leftrightarrow v_{pc1} \wedge v_{cc1})$ is stated as: *cc2(f1:F1, pc1:PC1, cc1:CC1, New_active4).*

In order to search the consistent values for the active variables, a predicate named *find_solution/1* is defined. This predicate recursively invokes a built-in predicate *indomain/1* to instantiate an active variable to a value of its domain. The code for this predicate is:

```
find_solution([]):- true.
find_solution([_:D|Active1]):-indomain(D), find_solution(Active1).
```

After programming all the variables, the domain for these variables, the constraints and the definitions for the problem specific predicates in a source file, the source file was compiled in the ECLiPSe system. Then, a query was executed to fire the problem solving process as shown in figure 5-5. The system found a configuration result which was shown in the output and error message section. Meanwhile, it indicated that there were more solutions. Continuing to execute the query, another configuration was found. Finally, two configurations for this oil pump body family were found by the system. Figure 5-6 shows the initial status of all the variables. The result for the first configuration is shown in figure 5-7 in which the variable - *HC3_1* is not active. The result for the second configuration is shown in figure 5-8 where the inactive variables include *PC2_1*, *PC4_1*, *PC4_2* and *CC4_1*. These results are consistent with the configurations of the two oil pump body variants in the example mentioned in chapter 2:

Configuration 1:

- Function modules: f1, f2, f4;

- Feature clusters: pc1, cc1, hc2, pc2, pc4, cc4;
- Feature variants: pc1_2, cc1_2, hc2_2, pc2_1, pc4_1, pc4_2, cc4_1;
- RMOPs: rm1, rm2, rm3, rm4, rm6, rm7;
- Precedence relationships: pr1, pr2, pr4, pr5, pr6.

Configuration 2:

- Function modules: f1, f2, f3;
- Feature clusters: pc1, cc1, hc2, hc3;
- Feature variants: pc1_1, cc1_1, hc2_1, hc3_1;
- RMOPs: rm1, rm2, rm4, rm5;
- Precedence relationships: pr1, pr2, pr3.

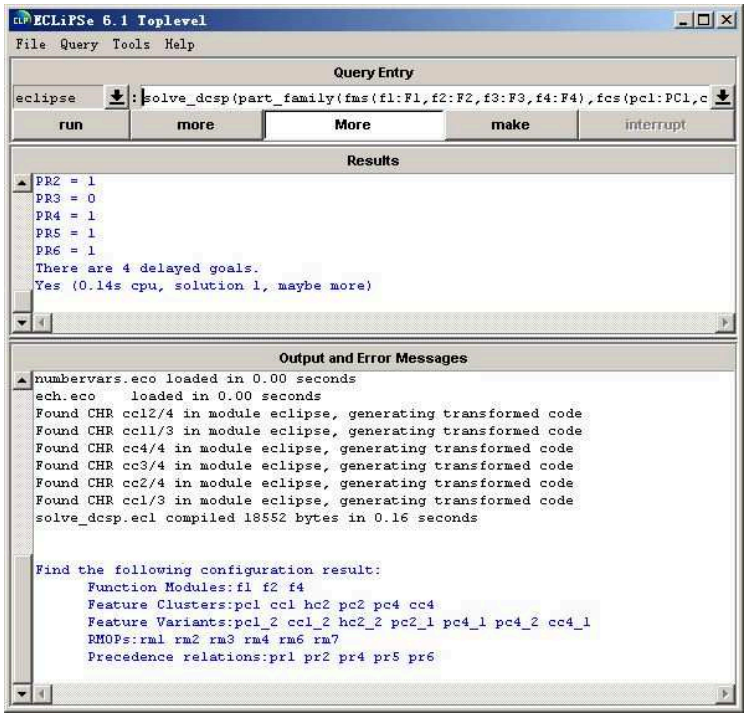


Figure 5-5: Result found after executing the query

Term display for solve_dcsp@eclipse

	1	2	3	4	5
1	F1{[0, 1]}	F2{[0, 1]}	F3{[0, 1]}	F4{[0, 1]}	PC1_1{[0, 1]}
2	PC1_2{[0, 1]}	CC1_1{[0, 1]}	CC1_2{[0, 1]}	HC2_1{[0, 1]}	HC2_2{[0, 1]}
3	PC2_1{[0, 1]}	HC3_1{[0, 1]}	PC4_1{[0, 1]}	PC4_2{[0, 1]}	CC4_1{[0, 1]}
4	RM1{[0, 1]}	RM2{[0, 1]}	RM3{[0, 1]}	RM4{[0, 1]}	RM5{[0, 1]}
5	RM6{[0, 1]}	RM7{[0, 1]}	PR1{[0, 1]}	PR2{[0, 1]}	PR3{[0, 1]}
6	PR4{[0, 1]}	PR5{[0, 1]}	PR6{[0, 1]}	unused	unused

Continue | stop none | stop all | Update on ground | Kill display

Figure 5-6: Initial status of the variables in the DCSP

Term display for solve_dcsp@eclipse

	1	2	3	4	5
1	1	1	0	1	0
2	1	0	1	0	1
3	1	HC3_1{[0, 1]}	1	1	1
4	1	1	1	1	0
5	1	1	1	1	0
6	1	1	1	unused	unused

Continue | stop none | stop all | Update on ground | Kill display

Figure 5-7: Fist solution found by the system

Term display for solve_dcsp@eclipse

	1	2	3	4	5
1	1	1	1	0	1
2	0	1	0	1	0
3	PC2_1{[0, 1]}	1	PC4_1{[0, 1]}	PC4_2{[0, 1]}	CC4_1{[0, 1]}
4	1	1	0	1	1
5	0	0	1	1	1
6	0	0	0	unused	unused

Continue | stop none | stop all | Update on ground | Kill display

Figure 5-8: Second solution found by the system

5.2 Process plan optimization for a specific variant

In a dynamic manufacturing environment, the manufacturing situation has high probability and frequency to suffer from the changes like new product introduction, machine breaks down or production volume adjustment. Consequently, the manufacturing process plan in a manufacturing system may also need to be changed in different manufacturing situations. Therefore, the department of manufacturing process planning should be flexible enough to response those dynamics. One of the conventional methods is to generate a new process plan from scratch at each time when the changes are needed. This method has obvious disadvantages on computation time in comparison with those methods established on the resource of the existing process plans.

Since RPP embodies all the feasible manufacturing process plans for any product/part

variants in a product/part family, the modular process plan elements derived from integrated product/part configuration and RPP configuration contain all the necessary information for building the feasible process plans for a specific product/part variant. When there is a need to fit a new manufacturing dynamic, one is able to construct an optimal process plan from the process plan elements in the RPP for this new manufacturing situation instead of generating it from scratch. Therefore, RPP provides a flexible way of process planning to the manufacturing system to handle the dynamics. After the generation of the modular process plan elements for a specific product/part variant by applying integrated product/part configuration and RPP configuration, in the second application stage, the situation of the current manufacturing system is taken into account and the final and optimal process plan for this manufacturing system is generated to meet the optimization objectives.

Depending on the dynamics encountered by a manufacturing system, different optimization objectives and constraint conditions have to be defined for different manufacturing situations. An optimal process plan for one manufacturing situation may be not optimal in another manufacturing situation. In this thesis, we identify two manufacturing scenarios described as follows (Xia et al., 2016):

- Scenario 1: Production shifts from one product/part variant to another product/part variant in a flexible manufacturing system with small production volume. In this scenario, the dynamic comes from an emergent production shifting request to manufacture another product/part variant, but the production volume for this product/part variant is small. In order to guarantee a short lead time and control the manufacturing cost, the current production resource and production layout should be maintained the same for this new introduced product/part variant. Thus, the optimization objectives for optimal process plan selection are: 1) minimize the production lead time; and 2) select the process plan for current variants as similar as possible with that for the previous variant.

- Scenario 2: Production shifts from one product/part variant to another product/part variant in a reconfigurable manufacturing system with large production volume. Being different with scenario 1, in scenario 2, the manufacturing system is reconfigurable and the production volume for the new introduced variant is large. In this scenario, the lead time maybe not so important than the production cost. It is necessary to adjust current

production resource and manufacturing system configuration in order to manufacture the new introduced variant with a low cost. Thus, the optimization objectives for optimal process plan selection are: 1) minimize the cost of the manufacturing process plan; 2) minimize the cost of the manufacturing resource and system configuration.

At product level, this stage refers to the construction and optimization of assembly process plans for a new product variant in a specific manufacturing system; while at part level, it refers to the construction and optimization of machining process plans for a new part variant in a specific manufacturing system. In the following sections, the approaches for the constructions and optimizations at these two levels are developed respectively.

5.2.1 Optimization of assembly process plan for a new product variant

In mass customization, a new product variant can be derived from a product configuration system. By applying integrated product configuration and RAPP configuration, a set of modular AND/OR graphs are selected from the RAPP at the moment that a set of part variants for the new product variant are selected. Then, the common part families and optional part families in these modular AND/OR graph are replaced by the corresponding part variants so as to form the modular AND/OR graphs for this specific product variant. These modular AND/OR graphs are then assembled together to form a complete AND/OR graph for this product variant which represents all the feasible assembly plans for this new variant.

As shown in figure 5-9, the construction of a complete AND/OR graph from a set of modular AND/OR graphs can be done easily by following the processes below:

Step 1: Find the modular AND/OR graph which contains the largest number of part variants from common part families and common parts as the base of the complete AND/OR graph;

Step 2: Add the rest of modular AND/OR graphs each by each into the base with the guidance of the following processes:

Step 2.1: Find the replacement points inside the base for adding the modular AND/OR graphs, a replacement point is an interactive component set C_b inside the base such that the interactive component set C_r at the root level inside the modular AND/OR graph being currently added satisfies the condition that $C_b \subset C_r$;

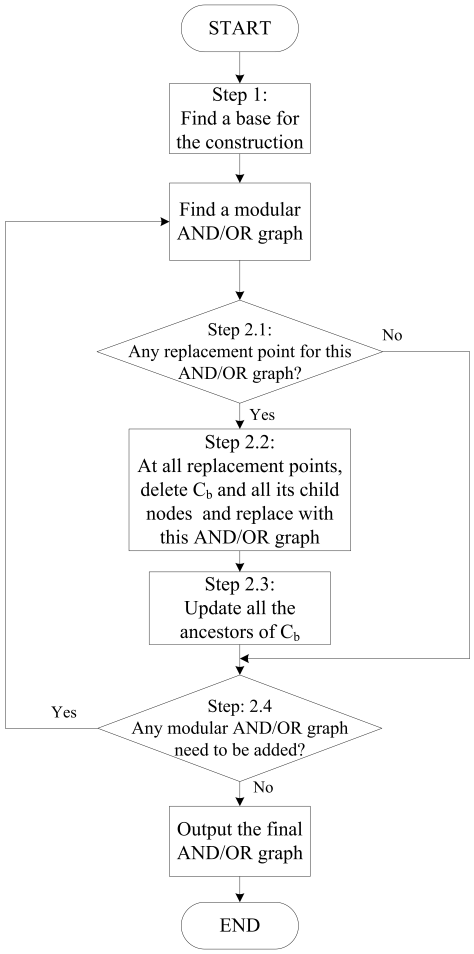


Figure 5-9: Process for the construction of AND/OR graph of a new configured product variant

Step 2.2: At each replacement point, delete C_b and all its child nodes, then replace with the modular AND/OR graph;

Step 2.3: Update the ancestors of C_b inside the base by uniting C_r into these nodes;

Step 2.4: The new AND/OR graph after adding the current modular AND/OR graph become the base for the next modular AND/OR to be added.

An illustration of the proposed process for the construction can be shown by using an example of the gear pump family. Suppose a gear pump variant with a pressure valve option is configured from the gear pump family and this example illustrates how to add the AND/OR graphs of the pressure valve to the base. Figure 5-10 shows the relevant modular process plans for this variant, which are configured from the RAPP of the gear pump family. For simplification, the modular process plans which are irrelevant to this

example are abbreviated because the combination of these modular process plans follow the same processes in this example. In figure 5-10, modular AND/OR graph 1 is the base for the construction of the complete AND/OR graph of this variant and only the interactive components sets related to pressure valve are shown; modular AND/OR graph 2 describes the assembly process plans for assembling the pressure valve on the head of the gear pump variant; and modular AND/OR graph 3 represents the assembly process plans for assembling the whole pressure valve. All these modular AND/OR graphs are instantiated from the RAPP of the gear pump family by replacing the variety components with the corresponding part variants.

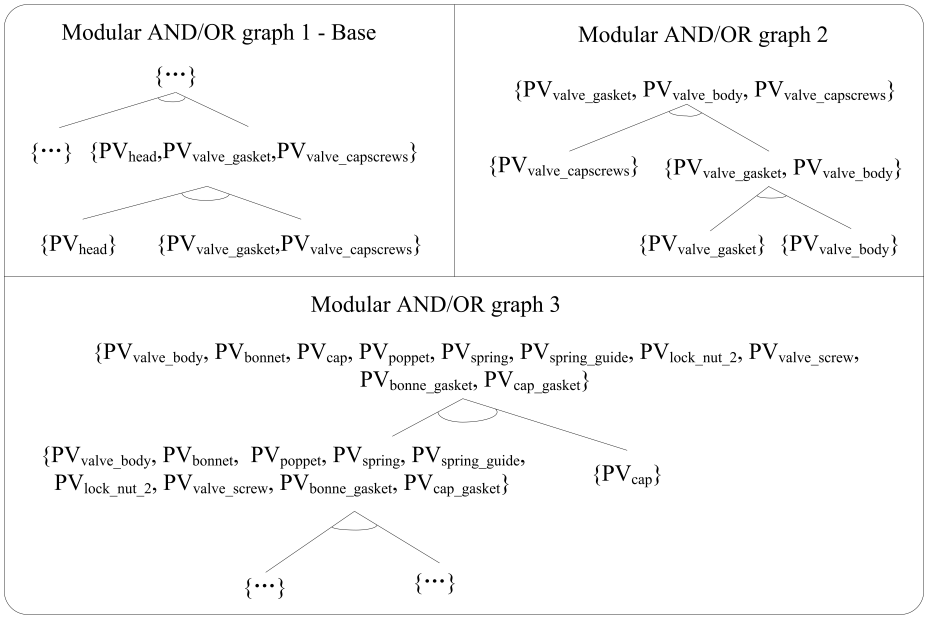


Figure 5-10: Modular AND/OR graphs for adding the AND/OR graphs of the pressure valve to the base

According to the step 2.1, because an interactive component set - { PV_{valve_gasket}, $PV_{valve\_capscrews}$} in the base is found to be a subset of the interactive components set - {PV_{valve_gasket}, $PV_{valve\_capscrews}$, PV_{valve_body}} at the root level of the modular AND/OR graph 2, modular AND/OR graph 2 is firstly added to the base and {PV_{valve_gasket}, $PV_{valve\_capscrews}$} is the replacement point for it. Figure 5-11 shows the new base after adding modular AND/OR graph 2. According to the step 2.2, {PV_{valve_gasket}, $PV_{valve\_capscrews}$} and all its child nodes are replaced by modular AND/OR graph 2; then in the step 2.3, all its ancestors are united with {PV_{valve_gasket}, $PV_{valve\_capscrews}$, PV_{valve_body}}. The new AND/OR graph formed after adding modular AND/OR graph 2 is used as the new base

for adding AND/OR graph 3 and the previous steps are repeated. In the new base, the replacement point for modular AND/OR graph 3 is found at the interactive components set - $\{PV_{valve\_body}\}$, because of $\{PV_{valve\_body}\}$ is a subset of $\{PV_{valve\_body}, PV_{bonnet}, PV_{cap}, PV_{poppet}, PV_{spring}, PV_{spring\_guide}, PV_{lock\_nut\_2}, PV_{valve\_screw}, PV_{bonnet\_gasket}, PV_{cap\_gasket}\}$. Therefore, $\{PV_{valve\_body}\}$ are replaced with modular AND/OR graph 3 and all its ancestors are updated. Figure 5-12 illustrates the combination between the new base and modular AND/OR graph 3.

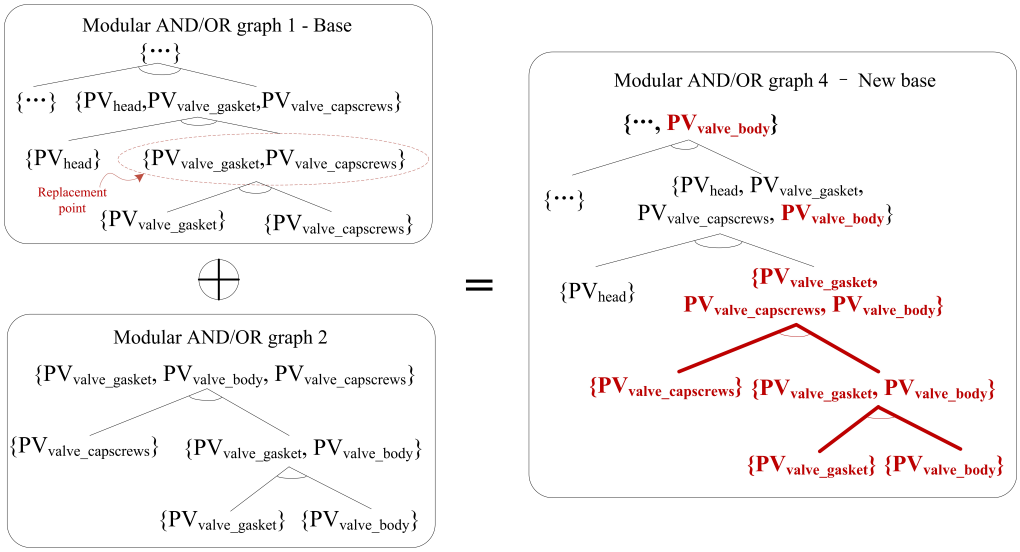


Figure 5-11: Adding modular AND/OR graph 2 to the base

A complete AND/OR graph contains all the feasible assembly process plans for a product variant. Thus, after generating the complete AND/OR graph for a product variant, the optimal assembly process plan can be found by applying a searching approach on this complete AND/OR graph. Generally, there are three categories of approaches to search the optimal assembly process plan from a set of feasible assembly process plans:

- Exact approaches without heuristics. The approaches in this group evaluate all the feasible assembly process plans, and then decide the most suitable one according to the proposed criteria. These approaches guarantee the optimum solution can be found, but they require large amounts of computation resource and computation time. These approaches could be enumerate algorithm and generate and test algorithm (Nievergelt, 2000).

- Exact approaches with heuristics. The approaches in this group apply heuristic strate-

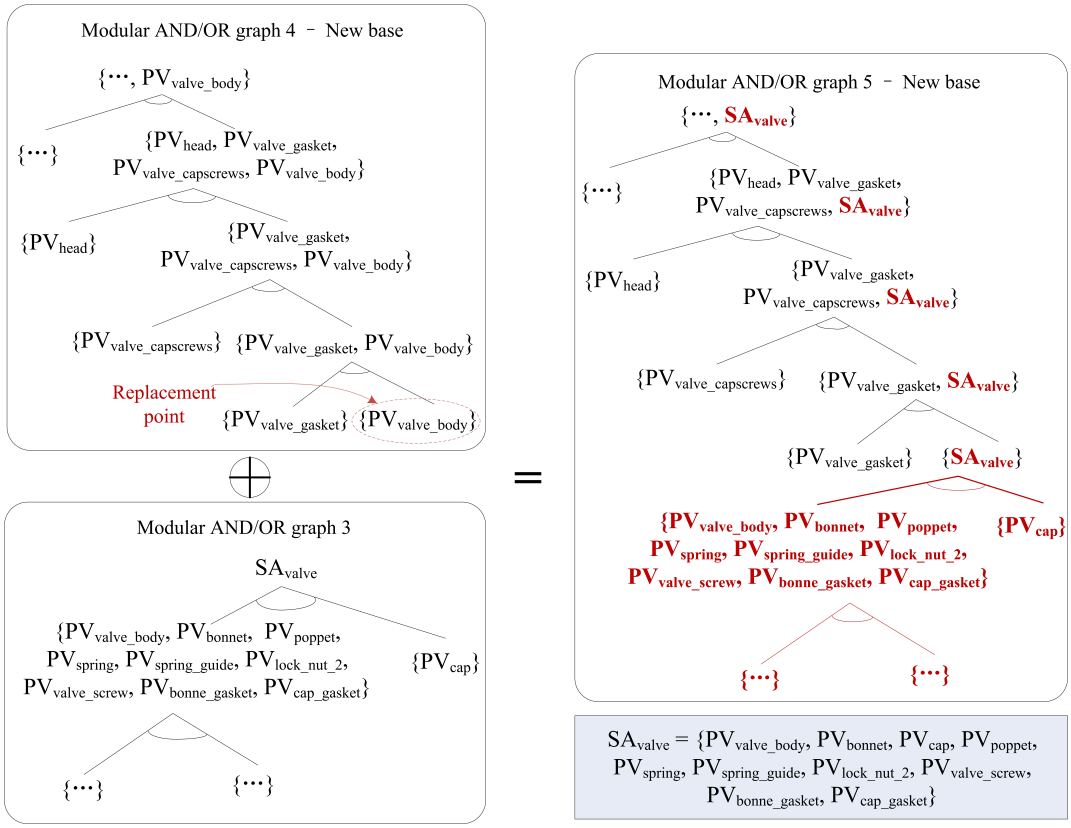


Figure 5-12: Adding modular AND/OR graph 3 to the new base

gies during the process of search space exploration. The heuristic strategies are able to guide the search towards the direction of the best solution, thus the search always stays in the most promising part of the solution space and avoids the less promising solution space to reduce computation time. It also guarantees to find the optimum solution. The examples could be best-first algorithm, AO* algorithm and branch-and-bound (Russell et al., 2003).

- Approximative approaches. The approaches in this group find an optimal solution without exploring the whole solution space. Thus, they have a certain degree of probability to find the best solution, but it is not guaranteed. Compared with exhaustive approaches, these approaches are much more efficient for a large problem. The examples could be genetic algorithm (Lazzerini and Marcelloni, 2000), hill climbing (Laperriere and ElMaraghy, 1996) and simulated annealing (Motavalli and Islam, 1997).

In this thesis, the following criteria are considered when choosing an optimization ap-

proach for searching the optimal assembly process plan:

- Because the feasible assembly process plans are represented in AND/OR graph, the approach should be able to process a AND/OR graph;
- The approach should perform better than exhaustive approach on computation time and cost;
- It is assumed that the optimization problem is moderate. In other words, it is assume that there is enough computation resource to find the optimum solution (not near optimum solution) within an acceptable time, therefore the approach should be able to guarantee a optimum solution.

Among these problem solving approaches for assembly process plan optimization, AO* algorithm is an effective approach for exploring an AND/OR graph which performs better than exhaustive approaches because of its heuristic estimate, and it can also guarantee to find the optimum solution as long as the algorithm satisfies the admissible condition. Therefore, AO* algorithm is used to search the optimal assembly process plan in the AND/OR graph of a product variant. Detailed explanation of AO* algorithm can be found in (Nilsson, 1980). AO* algorithm begins the search with the root node of the AND/OR graph and it gradually grows a search tree by expanding the visited nodes. During the search process, the algorithm always chooses the "most promising" candidate node for the further expansion. The most promising candidate node is a node which has the lowest cost among all the candidate nodes for the further expansion.

The cost of a node N in the search tree is calculated by using the following formula:

$$C(N) = \begin{cases} H(N) & \text{if } N \text{ is a leaf or has not been expanded} \\ \min\limits_{0<i<k} \left(\sum\limits_{j=1}^{m} (cost(N, N_{ij}) + C(N_{ij})) \right) & \text{if } N \text{ has been expanded.} \end{cases} \tag{5.1}$$

where $H(N)$ is a heuristic estimate of the contribution of the node N to the optimal solution and If N is a terminal node in the AND/OR graph, then $H(N) = 0$. $C(N)$ is the cost of N. $C(N)$ is calculated by using two different formulas according to the cases of the node N shown in figure 5-13. If N is a leaf or has not been expanded yet, then $C(N) = H(N)$; if N has been expanded, then $C(N) = \min\limits_{0<i<k} \left(\sum\limits_{j=1}^{m} (cost(N, N_{ij}) + C(N_{ij})) \right)$ where the subgraph

of N is divided into a set of groups - $\{N_1,\ N_2, ...,\ N_k\}$, each of which corresponds to one hyper-arc pointing from N to its successors $\{N_{k1},\ N_{k2}, ..,\ N_{km}\}$, N_{ij} is a child node of N in a subgraph group - N_i and $cost(N, N_{ij})$ is the cost of the arc pointing from N to N_{ij}.

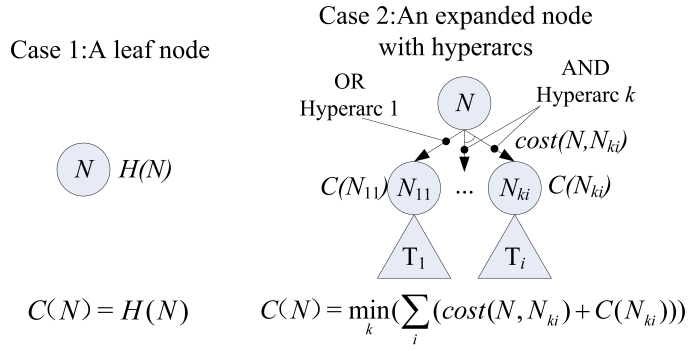


Figure 5-13: Illustration for the cost of the two cases of nodes

During the expansion of the solution tree, the algorithm updates the costs of the ancestors of the node currently being expanded. In order to keep the algorithm searching in the most promising direction, the ancestors with a OR subtree are checked at each time their costs are updated, once the cost of one of these nodes exceeds the cost of another candidate node, the algorithm leaves current subtree and goes to expand the subtree rooted at that candidate node.

In the two manufacturing scenarios mentioned in the beginning of section 5.2, the same AO* algorithm can be applied, but they use different ways to calculate the heuristic estimate - $H(N)$ of a node N and the cost of the arc - $cost(M, N)$ according to the different optimization objectives in the scenarios.

For scenario 1, as the optimization objectives are to minimize the production lead time and to maintain the APP as unchanged as possible compared with the APP of the previous variant, the calculations of $H(N)$ and $cost(M, N)$ should reflect the most promising node contributing to the optimal solution in terms of these two objectives. Therefore, we define the meaning of $H(N)$ and $cost(M, N)$ as follows in this scenario:

- $H(N)$ estimates that how different the final solution would be if the searching process expand the node N, compared with the AND/OR graph of the previous product.

- $cost(M, N) = \frac{1}{2} cost(M, \{N, N^{'}\})$, where M is a parent node of N and $cost(M, \{N, N^{'}\})$ describes the cost for choosing the assembly operation: $\{N, N^{'}\} \rightarrow M$ as a part of the

final solution. In the AND/OR graph, an AND hyperarc connects two child nodes, therefore $cost(M, N)$ is the half of $cost(M, \{N, N^{'}\})$.

To evaluate the heuristic estimate H for a component node N, we consider two factors:

- The number of the parts inside the component node $N : x$;
- The number of different parts between N and the corresponding node in the AND/OR graph of the previous product: n. The corresponding node of the component N is the most similar node at the same level in the AND/OR graph of the previous product as N in the AND/OR graph of current product.

Based on the two factors, the heuristic estimate H can be defined by using a function as follow:

$$H(N) = f(x, y) = \begin{cases} \frac{n}{x} & n \in \mathbb{Z}^+, x \in \mathbb{Z}^+, x \neq 1 \\ 0 & x = 1 \end{cases} \tag{5.2}$$

In order to guarantee the AO* algorithm to find the optimal solution, the heuristic estimate of a node N needs to satisfy the admissible condition:

$$H(N) \leq \min_{0<i<k} (\sum_{j=1}^{m} (cost(N, N_{ij}) + H(N_{ij}))) \tag{5.3}$$

where N_{ij} is a child node of N connected by a hyper-arc in a subgraph group - N_k and $cost(N, N_{ij})$ is the cost of the arc pointing from N to N_{ij}, $H(N_{ij})$ is the estimate of N_{ij}.

After determining the heuristic estimate of every node inside the AND/OR graph of current product, the value assignment for the cost of every arc have to satisfy the admissible condition. Here an example is used to show how to use the AO* algorithm to find the most similar assembly process plan for a new introduced product variant. Figure 5-14 shows the AND/OR graph for a new product variant. Figure 5-15 shows the assembly sequence for the previous product variant. The new variant shares a set of common components, a, b, c, d, f, g, with the previous product variant, but it has a new part - e and it does not include the part - i. In the AND/OR graph for the new variant, there are three feasible assembly sequences. By applying the AO* algorithm on this AND/OR graph, the most similar assembly sequence compared with the one in figure 5-13 can be found.

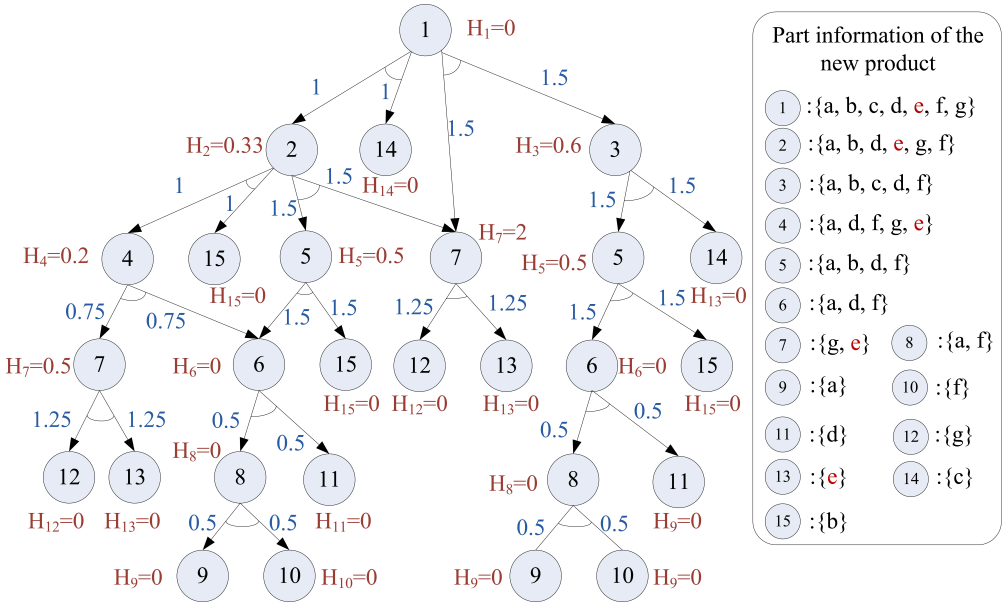


Figure 5-14: AND/OR graph for a new product variant

In the AND/OR graph of the new product variant, the nodes and the edges are labeled with the heuristic estimates and the costs respectively. The heuristic estimate for each node is calculated by using the formula 5.2. For example, for node 2 - {a, b, d, e, g, f}, the number of the parts in this component is 6 and the corresponding node in the assembly sequence of the previous product variant shown in figure 5-15 is {a, b, d, g, f, i}. By comparing the parts in node 2 and those in the corresponding node of node 2, the different parts between them consist of e and f, thus the value of n in the formula 5.2 is 2, and then the heuristic estimate for node 2 is 0.33. The cost for each edge is the half of the cost for the AND hyperarc where this edge belongs. The cost for a AND hyperarc is derived by considering the cost of the operation changes between the operation denoted by this AND hyperarc and the corresponding operation in assembly sequence of the previous variant. For example, depending on a specific case, the possible changes could be assembly tools change, assembly direction change and assembly force change.

The problem solving process of the AO* algorithm for searching the optimal assembly sequence of the new variant is shown in figure 5-16. The red edges mark out the most similar assembly sequence compared with the one in figure 5-15. The bold arrows indicate the searching directions of the AO* algorithm and the numbers labeled on them denote the searching sequences. These searching sequences are determined by comparing the cost estimates of all the possible searching direction. The AO* algorithm always chooses the most

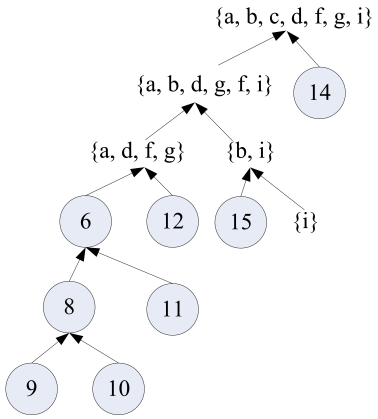


Figure 5-15: Assembly process plan for the previous product variant

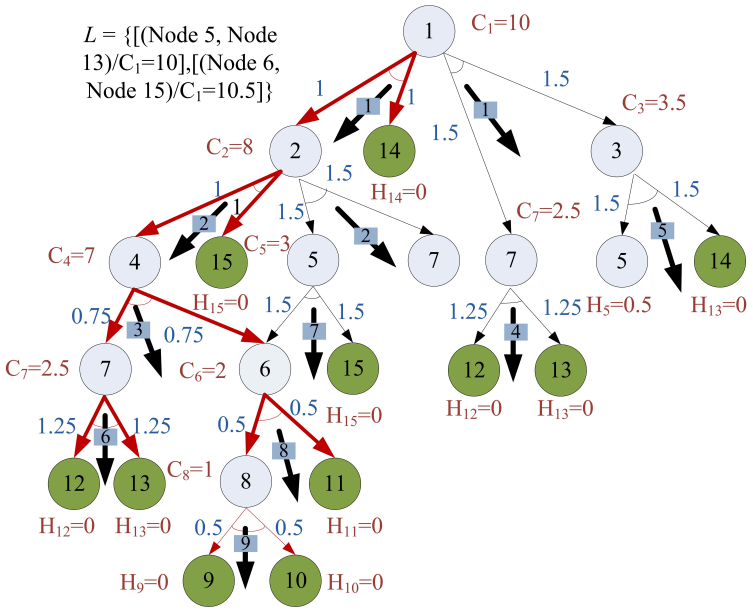


Figure 5-16: Problem solving process of the AO* algorithm for this example

promising direction with minimum cost estimates. After each extension has been made, the costs of all the ancestors of current node are updated. The algorithm terminates when all the leaf nodes of current subgraph being extended are marked as solved. As shown in the figure, the optimal assembly sequence is found after 9 times of irritations and the cost for this optimal assembly sequence is 10, while the minimum costs of the candidate subgraphs stored in L are equal or larger than 10. With the heuristic mechanism, the AO* algorithm can find the optimum solution without exploring the whole solution space, which consequently improves the searching efficiency.

For the scenario 2, it applies the same AO* algorithm to find the optimal assembly

sequence on the aspect of assembly cost and assembly time, but the ways to calculate the heuristic estimate of the nodes and the cost of the hyper-arcs in the AND/OR graph are different. In the scenario 2, the meaning of $H(N)$ and $cost(M, N)$ are defined as follows:

- $H(N)$ means the estimated assembly cost for the component denoted by node N. Three factors are considered when determining the value of $H(N)$: reorientation, manipulability and parallelism (Lee, 1991):

  - **Reorientation.** Generally, a single direction of assembly is considered more cost-effective than multiple directions of assembly. A single direction of assembly means all the other parts of a component are assembled from one single direction onto the base part of the component, while multiple directions of assembly means the base part needs to be reoriented to multiple assembly poses during the assembly of this component. The estimate on reorientation, $H_{\mathrm{r}}(N)$, can be calculated by using this formula:

$$H_{\mathrm{r}}(N) = \frac{n_{\mathrm{d}}}{n_{\mathrm{p}}} \tag{5.4}$$

    where:

    n_{d}: The number of different assembly directions of the parts in component N;

    n_{p}: The number of the parts excluding the base part in N.

  - **Manipulability.** Manipulability describes the difficulty degree for orienting and handling the component N. It is closely linked to the size, shape and weight of the component. The estimate on manipulability, $H_{\mathrm{m}}(N)$, can be calculated by firstly evaluating the orientation criteria and the handling criteria for measuring manipulability in table 5.2 and then using this formula:

$$H_{\mathrm{m}}(N) = \frac{S_{\mathrm{o}} + S_{\mathrm{h}}}{T_{\mathrm{o}} + T_{\mathrm{h}}} \tag{5.5}$$

    where:

    S_{o}: The score after evaluating the orientation criteria;

    S_{h}: The score after evaluating the handling criteria;

    T_{o}: The total score for orientation criteria;

    T_{h}: The total score for handling criteria.

Table 5.2: Criteria for measuring manipulability of a component(Lee, 1991)

Aspects of measurement	Criteria	Score
Orientation	Part tangles, nests or shingles	5
	Asymmetric part without marked polarities of weight or geometry	5
	Asymmetric part with marked polarities of weight or geometry	3
	Symmetric part	1
	Part delivered to the assembly station with a know orientation	1
	Total score -T_o:	15
Handling	Large off center weight potentially causing loss of orientation	5
	Very large parts	5
	Very small parts	5
	Fragile	3
	Flexible	3
	Irregular shaped part requiring special tooling	3
	Easily handled part with standard tooling	1
	Total score -T_o:	25

– **Parallelism.** An ideal assembly sequence should permit the subassemblies of a component are assembled in parallel to keep the intermediate stock at a low level. Parallelism describes the difficulty degree for a component to have parallel assembly sequences in its assembly sequence. The estimate on parallelism, $H_{\mathrm{p}}(N)$, could be calculated by using this formula:

$$H_{\mathrm{p}}(N) = \min_{0<i<k, i\in\mathbb{Z}} \frac{|n_i^a - n_i^b|}{n_c} \tag{5.6}$$

where:

k: The number of feasible cuts of the component N;

n_i^a: The number of parts in one subset V_a in the ith feasible cut;

n_i^b: The number of parts in anther subset V_b in the ith feasible cut;

n_c: The number of parts in the component N.

After determining $H_{\mathrm{r}}(N)$, $H_{\mathrm{m}}(N)$ and $H_{\mathrm{p}}(N)$ by using the formulas 5.4, 5.5 and 5.6 respectively, we can synthesize $H(N)$ by using this formula:

$$H(N) = \omega_{\mathrm{r}} H_{\mathrm{r}}(N) + \omega_{\mathrm{m}} H_{\mathrm{m}}(N) + \omega_{\mathrm{p}} H_{\mathrm{p}}(N) \tag{5.7}$$

where:

ω_{r}: the weight for the heuristic estimate in relation to reorientation;

ω_{m}: the weight for heuristic estimate in relation to manipulability;

ω_{p}: the weight for heuristic estimate in relation to manipulability;

$\omega_{\mathrm{r}} + \omega_{\mathrm{m}} + \omega_{\mathrm{p}} = 1$.

- $cost(M, N) = \frac{1}{2} cost(M, \{N, N'\})$, where M is a parent node of N and $cost(M, \{N, N'\})$ describes the cost for assembling M from its components, N and N'. The assembly cost of N and N' considers two factors: 1) difficulty degree to form M from assembling its components, N and N'; 2) assembly time. The approaches to evaluated the costs from these two aspects are proposed as follows:

  - Separability and reorientation of the components are considered as two criteria to evaluate the difficulty degree. Separability can be measured by the shape of the local depart space of the components N and N' which can be calculated by this formula (Mosemann et al., 1997):

$$cost_{\mathrm{s}}(M, \{N, N'\}) = 1 - \frac{Area[\prod(lds(N, N'), U)]}{4\pi} \tag{5.8}$$

    where:

    $lds(N, N')$: local depart space of N and N';

    $\prod(lds(N, N'), U)$: Projection operator which projects the local depart space onto a unit sphere U;

    $Area[\prod(lds(N, N'), U)]$: the area of the projection of the local depart space onto the unit sphere U.

    The formula used to calculate the reorientation cost is defined as follows:

$$cost_{\mathrm{r}}(M, \{N, N'\}) = \frac{\varphi}{\pi} \cdot \frac{g}{G} \tag{5.9}$$

where:

φ: reorientation angle;

g: mass of the reoriented component;

G: mass of M.

– The cost related to assembly time can be evaluated by this formula:

$$cost_{\mathrm{t}}(M, \{N, N'\}) = \frac{t_a}{t_m} \tag{5.10}$$

where:

t_a: the operation time to assemble N and N';

t_m: the minimum operation time among the possible assembly operations for M.

After calculating the costs related to difficulty degree and assembly time respectively, the total assembly cost of N and N' can be synthesized by using the following formula:

$$cost(M, \{N, N'\}) = \lambda(\omega_{\mathrm{s}} cost_{\mathrm{s}} + \omega_{\mathrm{r}} cost_{\mathrm{r}} + \omega_{\mathrm{t}} cost_{\mathrm{t}}) \tag{5.11}$$

where:

ω_{s}: the weight for the cost in relation to separability;

ω_{r}: the weight for the cost in relation to reorientation;

ω_{t}: the weight for the cost in relation to assembly time;

λ: the coefficient for $cost(M, \{N, N'\})$ to guarantee $cost(M, \{N, N'\}) \geq 2$.

5.2.2 Optimization of machining process plan for a specific variant

After application stage I, all the necessary process planning elements for a new part variant are derived from integrated part configuration and RMPP configuration, including all the machining features of this part variant and RMOPs related to these features as well as the precedence relationships between these machining features. In application stage II, these process planning elements are used to construct the optimal machining process plan for this new part variant with the consideration of a specific manufacturing scenario. In order to find the optimal machining process plan for a new part variant, the application in this stage is divided into four sequential steps as shown in figure 5-17.

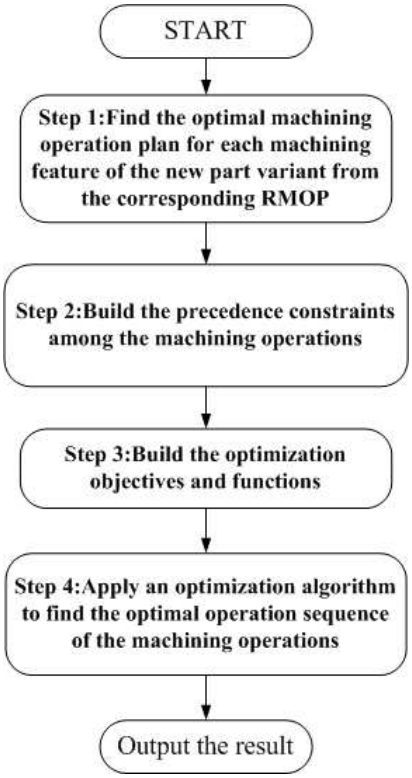


Figure 5-17: Four steps for machining process plan optimization in application stage II

Step 1: Find the optimal machining operation plan

Although the RMOP contains all the possible machining operation plans for any feature variant in a feature cluster, not every machining operation plan in the RMOP is qualified and cost-effective for the feature variants in a feature cluster. Some machining operation plans which are qualified for a feature variant could be unqualified or over-qualified for another feature variant in terms of their machining capabilities. Therefore, in the first step, we need to find the cost-effective machining operation plan for each feature variant of the new part variant from the corresponding RMOP.

The selection of the optimal machining operation plan follows three principles:

- Firstly, the machining operations in the optimal machining operation plan should be executable by current manufacturing system. This principle requires a machining operation plan in which the machining operations can be executed by the machines in the manufacturing system.

- Secondly, the machining capabilities of the optimal machining operation plan must satisfy all the machining requirements of the feature variant.

- Thirdly, the optimal machining process has the shortest path.

Since a RMOP is represented as a directed graph, finding the optimal machining operation plan in the RMOP can be solved as a shortest path problem. A best-first algorithm is developed to search the shortest path in the directed graph of a RMOP. The algorithm starts at the start node of the directed graph and continually expands the current node until a solution has been found. In each expansion, the algorithm always chooses the most promising successor to expand. This is guaranteed by computing a heuristic estimate for each candidate and the most promising one is the one that has minimum heuristic estimate.

The heuristic estimate in the algorithm is defined as follows:

$$f(N) = g(N) + h(N) \tag{5.12}$$

where:

$f(N)$: the heuristic estimate of a node N;

$g(N)$: the shortest path cost from the start node to node N;

$h(N)$: the cost estimate of an optimal path from node N to the goal node.

The shortest path cost $g(N)$ is calculated by accumulating the costs of the edges from the start node to node N in current path:

$$g(N) = \sum_{i=1}^{k} c(N_{i-1}, N_i) \tag{5.13}$$

where:

N_0: the start node in the directed graph;

N_k: current node N;

$N_1, N_2, ..., N_{k-1}$: the nodes passed by current path from N_0 to N;

$c(N_{i-1}, N_i)$: the edge cost between node N_{i-1} and node N_i.

The edge cost between node N_{i-1} and node N_i in formula 5.13 can be calculated by the following formula:

$$c(N', N) = \begin{cases} 1 & \text{if } N' \text{ and } N \text{ can be executed on the same machine} \\ 1.5 & \text{if } N' \text{ and } N \text{ cannot be executed on the same machine} \\ 100 & \text{if } N \text{ cannot be executed in the manufacturing system} \end{cases} \tag{5.14}$$

The cost estimate $h(N)$ can be calculated by using this formula:

$$h(N) = 1 - \frac{n_{\mathrm{sr}}}{n_{\mathrm{r}}} \tag{5.15}$$

where:

n_{sr}: the number of satisfied machining requirements at the node N;

n_{r}: the number of all the machining requirements of the feature variant.

After defining the heuristic estimate for the nodes, the best-first algorithm is developed to search the optimal machining operation plan with the help of the heuristic estimate. The whole procedure of the best-first algorithm is described by the pseudo code as follows.

Pseudo code of best first algorithm for searching the optimal MOP

```
/** A best first algorithm for searching the optimal machining operation plan in
    a directed graph**/
1 initialize OPEN=[N_s];
2 initialize CLOSED=[];
3 while OPEN≠ φ do {
4   find the node N_i with the least f(N_i) in OPEN
5   delete N_i from OPEN
6   generate the successors of N_i
7   for all N_k is a successor of N_i, do{
8      Predecessor(N_k) = N_i
9      calculate f(N_k)
10     if (N_k is the goal node){
11        return current path and stop the search}
12     else if (there exists a N_k in OPEN which has a lower f(N_k)){
13        skip this N_k}
13     else if (there exists a N_k in CLOSED which has a lower f(N_k)){
14        skip this N_k}
15     else {add N_k into OPEN}}
16   add N_i into CLOSED}
```

Here we use an example to illustrate the searching process of the proposed best first algorithm. Assume that there is a hole feature variant on a new part variant and its machining requirements are listed in table 5.3.

Table 5.3: Machining requirements of a hole feature variant

Surface roughness ($R_{sr}/\mu m$)	Geometrical specifications		Tolerance		
	Diameter (R_{dia}/mm)	Depth (R_{dept}/mm)	Dimensional tolerance(R_{dt})	True position (R_{tp}/mm)	Cylindricity (R_{cy}/mm)
0.3	$\phi 10$	50	$IT6$	0.04	0.005

In addition, the RMOP for this hole feature variant is shown in figure 5-18 and we know there is a CNC machine which can perform all the operations in this RMOP.

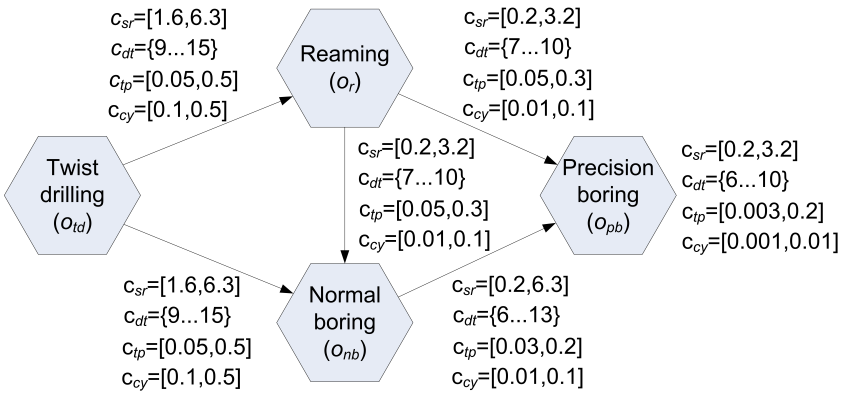


Figure 5-18: RMOP for a hole cluster

Figure 5-19 shows the searching process of the proposed algorithm for this example. The process starts at the first node O_{td}, and then the algorithm continues to find its two successors, O_{nb} and O_r. For the machining operation O_{nb}, the cost estimate $h(O_{\mathrm{nb}})$ is 0.25, because it satisfies 3 machining requirements out of the 4 requirements, while the cost estimate of O_{r}, $h(O_{\mathrm{r}})$, is 0.75, because the operation O_{r} only satisfies 1 machining requirement out of 4 requirements. Because a CNC machine can perform all the operations in this RMOP, $c(O_{\mathrm{td}}, O_{\mathrm{r}})$=$c(O_{\mathrm{td}}, O_{\mathrm{nb}})$=1. Then, the heuristic estimates for these two operations can be calculated by adding the edge costs to the cost estimates, which derives $f(O_{\mathrm{nb}})$=1.25 and $f(O_{\mathrm{r}})$=1.75. Comparing the heuristic estimates of O_{nb} and O_{r}, the algorithm chooses O_{nb} as the node for expansion. In the next expansion, the goal node O_{pb} that satisfies all the machining requirements is encountered. Thus, the algorithm stops at the goal node and returns the current path: $O_{\mathrm{td}} \rightarrow O_{\mathrm{nb}} \rightarrow O_{\mathrm{pb}}$. This path is the optimal machining operation plan for this hole feature variant.

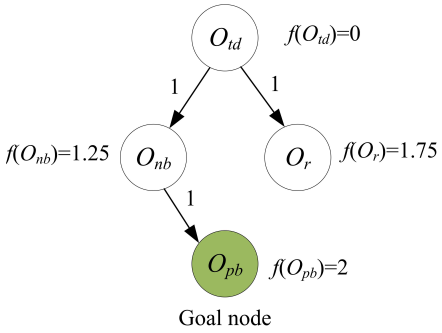


Figure 5-19: Searching process of the best first algorithm for a hole feature variant

Step 2:Build the precedence constraints among the machining operations

In step 1, the optimal machining operation plans are found for all the feature variants of the new part variant. These machining operation plans are then reconstructed into an optimal machining process plan for the new part variant considering the corresponding optimization objectives. For this purpose, in step 2, the precedence constraints among the machining operations in the machining operation plans are built. The precedence constraints are those constraints that the machining operations must satisfy. They come from two aspects:

- Feature precedence relations. These relations are derived after performing integrated part configuration and RMPP configuration. In this step, these precedence relations between features are mapped to the constraints between the machining operations used to machine these features. As mentioned in chapter 3, section 3.2, five types of feature precedence relations are defined. Therefore, different rules are defined to map these five types of precedence relations to the operation constraints. Assume that fv_1, fv_2 are two feature variants; op_1 is the machining operation plan for fv_1; op_2 is the machining operation plan for fv_2; O_{s1} is the first operation in op_1 and O_{e1} is the last operation in op_1; O_{s2} is the first operation in op_2 and O_{e2} is the last operation in op_2; $position(O)$ represents the position of operation O in the final machining operation plan. Then, we have the following rules:

  Rule 14. If fv_1 *softBefore* fv_2, then $position(O_{s1}) < position(O_{s2})$;

  Rule 15. If fv_1 *hardBefore* fv_2, then $position(O_{e1}) < position(O_{s2})$;

  Rule 16. If fv_1 *softImmeBefore* fv_2, then $position(O_{s2}) = position(O_{s1}) + 1$;

  Rule 17. If fv_1 *hardImmeBefore* fv_2, then $position(O_{s2}) = position(O_{e1}) + 1$;

Rule 18. If fv_1 *equal* fv_2, then $position(O_{s1}) = position(O_{s2})$ and $position(O_{e1}) = position(O_{e2})$.

- Machining operation plan. A machining operation plan also defines the precedence of the machining operations. In order to map the operation precedence in the machining operation plans to the precedence constraints for the machining process plan, we define the following rule:

  Rule 19. If O_1 is a directed predecessor of O_2 in a machining operation plan, then $position(O_1) < position(O_2)$.

Step 3: Build the optimization objectives and functions

In the optimal machining process plan, the sequence of the machining operations must not only satisfy all the precedence constraints built in step 2, but also meet the optimization objectives according to a specific manufacturing scenario. In this step, the optimization objectives for machining operation sequencing and the functions to evaluate the solution candidates for these optimization objectives are built. As mentioned at the beginning of section 5.2, two manufacturing scenarios are identified for the applications: 1) production shift in a flexible manufacturing system with low production volume; 2) production shift in a reconfigurable manufacturing system with high product volume.

In the first scenario, the optimal machining process plan for the new part variant should be as similar as possible to the machining process plan of the previous part variant. In addition, since the production shift is an emergent production order, the machining lead time is another optimization objective. Based on these two optimization objectives, an objective function for this scenario is defined as:

$$F = \omega_s(1 - F_s) + \omega_t F_t \tag{5.16}$$

where:

F_s :similarity coefficient between the MPP candidate and the MPP of the previous part variant;

F_t : total machining time for the MPP candidate;

ω_s : weight for the similarity coefficient;

ω_t : weight for the total machining time.

In formula 5.16, ω_s+ω_t=1. The optimal MPP in this scenario is the one has the minimal value of F. In the literature, many effective methods has already been proposed to evaluate F_s and F_t (Goyal et al., 2013; Ding et al., 2005). Therefore, the methods to calculate F_s and F_t are not main focuses of this thesis.

In the second scenario, machining cost and machining time are the two optimization objectives because a reconfigurable manufacturing system has strong adaptability to a MPP variant. Therefore, the objective function for the manufacturing scenario 2 is defined as follows:

$$F = \omega_c F_c + \omega_t F_t \tag{5.17}$$

where:

F_c: total cost for the MPP candidate;

F_t: total machining time for the MPP candidate;

ω_c: weight for the total cost;

ω_t: weight for the total machining time.

Similarly, in formula 5.17, $\omega_c + \omega_t$=1. The optimal MPP in this scenario is the one has the minimal value of F. For evaluating the total cost of a MPP, there are also many methods proposed in the literature(Li et al., 2002; Liu et al., 2013), therefore, it is not be detailed in this thesis.

Step 4: Apply an optimization algorithm to find the optimal operation sequence

The final step is to use an optimization algorithm to find the optimal operation sequence which results a minimal value for the objective function defined in step 3, meanwhile satisfies all the precedence constraints defined in step 2. For a small sequencing problem with a small number of operations, a deterministic algorithm can be applied and the optimum process plan can be found; for a large sequencing problem with large number of operations, a stochastic method is more suitable than a deterministic one, but the solution found is a near-optimum solution. Many optimization approaches are available in the literature, for example, genetic algorithm (Qiao et al., 2000), ant colony algorithm (Liu et al., 2013), simulated annealing (Ma et al., 2000) and hybrid algorithm (Li et al., 2002). Therefore, this step is not be detailed in this thesis.

5.3 Conclusions

This chapter answers the question that how to apply RPP to handle the manufacturing complexity induced by product/part variety. The application of RPP consists of two stages. In the first stage, product/part configuration and process configuration are integrated together. The integration of product/part configuration and RPP configuration is established on the modular representations of product/part variety model and RPP models. A product/part variant and all the necessary process plan elements for building the process plans of this product/part variant are generated together by integrated product/part configuration and RPP configuration. In this way, the process planning efficiency of a product/part family is improved significantly because the process plans for any variant in the family are able to be constructed from a set of process plan components instead of generating them from scratch.

In the second stage, those process plan elements derived from integrated product/part configuration and RPP configuration are used to build the feasible and optimal process plan for the desired product/part variant. The optimal process plan is tailored to a specific manufacturing system with a consideration of the available manufacturing resources and the corresponding optimization objectives. In this stage, we see that RPP provides enough flexibility to handle the dynamics from the manufacturing system because the process plan elements configured from RPP can be used to construct all the feasible process plans of a product/part variant, once the situation of the manufacturing system changes, the optimal process plan adapting to the changes can be identified from the feasible process plans of this product/part variant.

A DCSP-based approach is developed for realizing integrated product/part configuration and RPP configuration. In this approach, the modular product/part variety components defined in the feature-based variety model and the modular process planning components defined in the RPP model are mapped as the variables in a DCSP. If necessary, the attributes of some variety components can also be mapped as the attribute variables in the DCSP. The values for the component variables are limited in a Boolean domain. If a component variable takes a value of "1", then the component denoted by this variable is chosen in the configuration solution; if it takes a value of "0", then the component is not chosen in the final solution. The value for the attribute variables could be limited in a set or an interval.

Two kinds of configuration constraints are defined in the DCSP that are compatibility

constraints and activity constraints. Compatibility constraints restrict the selectivity relations among the configurable components. They are already defined in the feature-based product/part variety model with the propositional-logic-based representation scheme. Activity constraint is a part of a special mechanism in DCSP which distinguishes it from the conventional CSP. An activity constraint defines the conditions under which a variable may or may not be actively considered as a part of a final solution. Through these activity constraints, the solution process only assigns values to the variables which are relevant to the final solution. This special mechanism reduces the number of variables and compatibility constraints that need to be processed during the problem solving process. Compared with conventional CSP, this mechanism can improve the problem solving efficiency when there are large numbers of variables and constraints.

Two manufacturing scenarios are identified for the application stage II. Each manufacturing scenario has its own optimization objectives. The approaches for APP optimization and MPP optimization in the two manufacturing scenarios are investigated respectively. For APP optimization, the modular AND/OR graphs for a specific product variant are firstly assembled together to form a complete AND/OR graph for this product variant. The feasible APPs for this product variant are included in the complete AND/OR graph. Then, an AO* algorithm is applied to search the optimal APP for the product variant. According to the defined optimization objectives, the methods to calculate the heuristic estimate and the edge cost are proposed in order to guide the algorithm to keep searching the most promising nodes. Different functions to calculate the heuristic estimate and the edge cost have been proposed according to the different manufacturing scenarios.

For MPP optimization, a four-step approach is proposed. In the first step, the optimal machining operation plan for each feature variant of the new part variant is generated by using a best-first algorithm; then, in the second step, the precedence constraints among the machining operations in the derived operation plans are built according to the feature precedence and their precedence in the operation plans; in the third step, the optimization objectives and objective functions for the two manufacturing scenarios are defined; based on the information defined in the previous steps, in the fourth step, an optimization algorithm is applied to the machining operations to find the optimal operation sequence which meets the optimization objectives. Because many objective functions for computing the machining cost and machining time have been put forward in the literature as well as the optimization

algorithms, the third step and the fourth step are not detailed in this thesis.

The proposed approaches and algorithms can be implemented into computer programs for automating the two application stages. An implementation for automating integrated part configuration and RMPP configuration has been done. In the CSP approach for integrated product/part configuration and RPP configuration, only set domain and Boolean domain are considered for the configuration variables. The Boolean-CSP approach needs to be extended with interval CSP solver to support the configuration variables with intervals as their domains. Although the implementations for the algorithms in optimization of AP-P/MPP are not shown in this thesis, the proposed algorithms and approaches are followed by illustrative examples to show their feasibility.

Conclusions, Limitations and Perspectives

Conclusions

In current manufacturing paradigms, conventional process planning approaches are very inefficient to handle the process planning complexity induced by product variety and manufacturing dynamics, because they treat each product/part variant individually. RPP is a new CAPP approach which targets to the generation of process plans for a product/part family instead of that for one single product/part.

The research reported in this thesis gives major contributions to the methodology, architecture, representation models and algorithms for reconfigurable process planning at two granularity levels: product family and part family. In this thesis, a product/part family is defined as a domain of product/part variants where sets and intervals are used to represent the attribute values of their design characteristics. By this way, the number of variants in a product/part family could be infinite. Figure 6-1 shows the global framework of the proposed approaches for RPP.

Two kinds of principle representation models are defined in this framework. Feature-based product/part variety model represents the design specifications of a product/part family in a structured and modular way. It provides the necessary data for generating RPP as well as the information for configuring a product/part variant. RPP models, including RAPP and RMPP, provide graph-based methods to represent modular process plans for a product/part family.

With modular technology, platform-based technology and configuration technology in mind, a reconfigurable process plan is defined as a set of modular process plan components

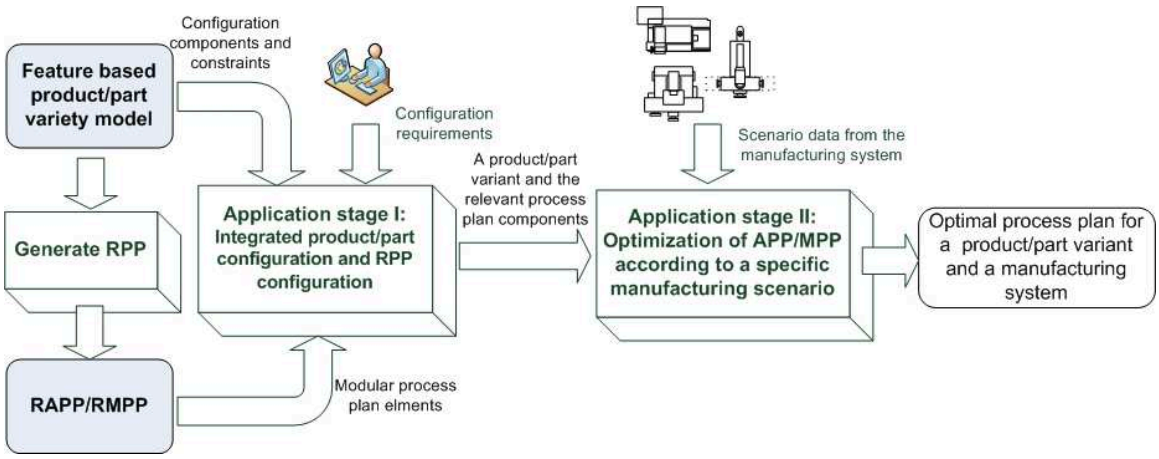


Figure:6-1: Framework of reconfigurable process planning for the two granularity levels

from which all the feasible process plans for any variants in a product/part family can be constructed. Reconfigurable process plan can be considered as an intermediate process plan which reflects the commonality among the process plan variants of all the product/part variants in the family. The reconfigurable process plan for a product/part family can be generated as soon as the design specifications of the product/part family are available. The final process plan for a product/part variant can be configured from the reconfigurable process plan once the configuration of this variant and the manufacturing scenario are specified.

In addition, in this framework, three kinds of model handling mechanisms are defined in order to automatically process the information represented by both product/part variety model and RPP models. RPP generation mechanism analyzes the process-related information in the product/part variety model of a product/part family in order to generate the reconfigurable process plan. Integrated product/part configuration and RPP configuration selects the compatible product/part components for a product/part variant according to a set of product/part configuration requirements. Meanwhile, it also selects the modular process plan components from the RPP corresponding to the selected product/part components. It is the first application stage of RPP. The modular process plan components derived from the RPP are the building blocks for the complete process plans of a product/part variant. Based on this, in the second application stage of RPP, mechanisms are proposed to generate the optimal process plan for a product/part variant meeting the optimization objectives of a specific manufacturing scenario.

In order to automate the representation models and the model handling mechanisms in

the framework, this thesis proposes the following models and approaches:

- **For feature-based product variety model**
  - Feature-based product variety decomposition architecture which represents the functional structure and the physical structure of a product family in a modular way;
  - A datum-flow-chain-based representation mechanism which represents the mating information between the mating parts at two levels: product family level and product variant level;
- **For feature-based part variety model**
  - A feature-based part variety decomposition network which represents both the functional and the physical structures of a part family in a modular way. Three variety levels are defined in this model: function module level, feature cluster level and feature variant level;
  - A knowledge-based representation mechanism to express the feature interactions between the feature variants for one single part variant, and a set-based approaches to express the interactions between the feature clusters for a part family;
- **For both product variety model and part variety model**
  A propositional-logic-based scheme to represent the configuration relations among the variety components in the variety model of a product/part family;
- **For RPP representation model at product level**
  - A concept of RAPP, which is defined as a set of modular assembly process plans satisfying all the assembly requirements of a group of interactive product variety components. The group of interactive product variety components could be either a mating module at product family level or a mating module at product variant level;
  - An AND/OR-graph-based representation model for RAPP, in which a RAPP is represented as a set of modular AND/OR graphs for a mating module;
- **For RPP representation model at part level**

  - A concept of RMOP, which is defined as a set of similar machining operation plans that satisfy all the machining requirements of all the feature variants in a feature cluster;
  - A directed-graph-based representation model for RMOP. All the feasible machining operation plans for any feature variant in a feature cluster can be found from the directed graph model for the RMOP of this feature cluster;
  - A concept of RMPP, which is defined as a combination of the RMOPs corresponding to all the feature clusters of a part family and the precedence relations between the interactive part variety components;

- **For RAPP generation mechanism**

  A cut-set-based disassembly approach to generate the AND/OR graphs for all the mating modules defined in the representation model of a product family. A general algorithm has been proposed to automate the cut-set-based disassembly approach and a guided Karger's algorithm has been developed to automatically generate all the feasible cuts for a reduced graph;

- **For RMPP generation mechanism**

  - A knowledge-based approach to select the feasible machining operations for a feature cluster. In this approach, a knowledge base has been developed to organize the domain knowledge on machining operation selection. This knowledge is expressed in first order logic language. In addition, a resolution-based breadth-first algorithm has been developed to search all the feasible machining operations for a feature cluster supported by the knowledge in the knowledge base;
  - A depth-first operation sequencing algorithm to find all the possible sequences of the selected machining operations, which explicitly form the RMOP for a feature cluster;

- **For integrated product/part configuration and RPP configuration**

  A DCSP-based approach to find the solutions for both the configuration of a product/part variant and the configuration of the modular process plan elements in RPP. By

using the active mechanism in the approach, the number of variants explored during the problem solving process can be reduced dynamically;

- **For construction and optimization of the final assembly process plan**
  - An approach to construct the complete AND/OR graph for a product variant based on the modular AND/OR graphs derived by integrated product configuration and RAPP configuration;
  - Cost estimate methods for AO* algorithm to guide the algorithm to find the optimal assembly process plan in a AND/OR graph considering the different criteria as optimization objectives for different manufacturing scenarios;
- **For construction and optimization of the final machining process plan**
  - A four-step approach to generate the optimal machining process plan for a part variant based on the modular process plan elements derived by integrated part configuration and RMPP configuration;
  - A best-first algorithm to find the optimal machining operation plan from a RMOP for a feature variant.

In order to illustrate the proposed representation models, a gear pump family and an oil pump body family are used as examples. The detailed information of these two examples is detailed in the Appendix. For all the algorithms proposed in this thesis, examples are followed to show their results and feasibility. In present stage, only a part of implementations have been done, including the depth-first algorithm for machining operation sequencing, the DCSP-based approach for integrated product/part configuration and RPP configuration, and the best-first algorithm to search the optimal machining operation plan from a RMOP. In the future work, the proposed models, approaches and algorithms should all be implemented into computer programs and these programs should be integrated into the RPP system depicted by the framework in figure 6-1.

Compared with conventional CAPP approaches, the RPP approaches proposed in this thesis result the following two main advantages:

- Reduce the process planning complexity for a product/part family

  This advantage is guaranteed by the principle of modularity and commonality in RPP.

If we consider a product family consisting of n product variants and we divide the process planning efforts for one product variant into two portions: 1) the efforts to analyze the information which is common in every product variant, such as generating the assembly process plan for a common subassembly; 2) the efforts to analyze the information which is not common in every product variant. The first portion is denoted as e^1 and the second one is denoted as e_k^2, then we use E_1 and E_2 to denote the total efforts for conventional CAPP approaches and RPP approach respectively and we have:

- For the conventional CAPP approaches, because they treat the variants individually, $E_1 = ne^1 + \sum_{k=1}^{n} e_k^2$;
- For the RPP approach proposed in this thesis, because modularity and commonality are considered for process planning, $E_2 = e^1 + \sum_{k=1}^{n} e_k^2$;

It is obvious that $E_2 < E_1$, so the RPP approach reduces the process planning complexity compared with conventional CAPP approaches.

- Improve the process planning flexibility for the dynamics in manufacturing system
  This advantage is guaranteed by the platform-based technology and the configuration-based technology. In conventional CAPP approaches, when the manufacturing scenario is changed, the optimal process plan for the new scenario has to be regenerated from scratch. However, in RPP, the optimal process plans for different manufacturing scenarios are derived from the same set of process plan components by using the same optimization approach with different objective functions.

Limitations

The proposed approaches in this thesis have the following limitations:

- The proposed approaches in this thesis are only applicable for the product/part variants whose design specifications have been encompassed in the product/part variety model. For a new product/part variant which contains newly introduced features, these new features have to be merged into the product/part variety model before the proposed approaches can be applied.

- The assembly process plan considered in this thesis is limited to a sequential, non-linear and monotone assembly process plan. (An assembly process plan is sequential if it can be decomposed into a sequence of assembly operations such that in each operation only one element is added. A non-linear assembly process plan is an assembly sequence which allows a subassembly to be added once at a time. In a monotone assembly process plan, each part is inserted immediately into its final position.)

- The proposed process planning approaches limit themselves to conceptual process planning in which process selection and process sequencing are their main focuses. Therefore, this thesis does not refer to tool selection, machine selection and process parameter and NC program generation.

- In the CSP approach for integrated product/part configuration and RPP configuration, only set domain and Boolean domain are considered for the configuration variables. The Boolean-CSP approach needs to be extended with interval CSP solver to support the configuration variables with intervals as their domains.

- In current stage, the configuration constraints for integrated product/part configuration and RPP configuration are generated manually according to the proposed generation guidelines, but this repetitive activity can be automated in the future.

- Considering the existing optimization approaches in the literature, this thesis only investigate the optimization objectives and functions for searching the optimal machining process plan for a specific part variant, thus, the implementations for process plan optimization are not detailed in this thesis.

Perspectives

Reconfigurable process planning is an emerging process planning approach, which has the capabilities to handle the process planning complexity brought by product variety and manufacturing dynamics. It is a key enabler for current manufacturing paradigms: MC and P3S. However, very few studies at present have chosen it as their research focuses in the literature. Reconfigurable process planning still exists at the conceptual and theoretical level. This thesis explores the concepts, models and approaches for RPP and tries to push RPP towards practice. To light the future work, the following issues could be considered:

- Further development of the cost estimate for the AO* algorithm to make it closer to the real cost of the nodes in the searching graph in terms of the optimization objectives.
- Investigation of more application scenarios for RPP. For example, in an application scenario where the machine layout is configurable, the optimization of process plan should consider the optimization of the machine layout. In another word, an interesting future work could be integrated process plan configuration and manufacturing system configuration.
- Validation in an industrial case study. RPP should be tested in an industrial case study to show its advantages compared with the conventional CAPP approaches;
- Automation, to a certain extent, of the generation of configuration constraints for integrated product/part configuration and RPP configuration. The configuration constraints come from the configurable relations between the variety components defined in product/part variety model and also the mapping relations between the product/part components and the process components in RPP model. The generation of these configuration constraints involves large amounts of repetitive work, which can be automated.

Bibliography

Agrawal, R., Shukla, S., Kumar, S., Tiwari, M., 2009. Multi-agent system for distributed computer-aided process planning problem in e-manufacturing environment. The International Journal of Advanced Manufacturing Technology 44 (5-6), 579–594.

Aldanondo, M., Vareilles, E., 2008. Configuration for mass customization: How to extend product configuration towards requirements and process configuration. Journal of Intelligent Manufacturing 19 (5), 521–535.

Allada, V., Anand, S., 1995. Feature-based modelling approaches for integrated manufacturing: state-of-the-art survey and future research directions. International Journal of Computer Integrated Manufacturing 8 (6), 411–440.

Amaitik, S., Kilic, S., 2005. Step-based feature modeller for computer-aided process planning. International Journal of Production Research 43 (15), 3087–3101.

Azab, A., ElMaraghy, H., Samy, S., 2009. Reconfiguring process plans: A new approach to minimize change. In: Changeable and Reconfigurable Manufacturing Systems. Springer, pp. 179–194.

Azab, A., Perusi, G., ElMaraghy, H., Urbanic, J., 2007. Semi-generative macro-process planning for reconfigurable manufacturing. In: Digital Enterprise Technology. Springer, pp. 251–258.

Barrabes, M., Villeneuve, F., 1993. Object data base, ai and cad-cam: application to the process ascending generation (pag) concept. In: CompEuro'93.'Computers in Design, Manufacturing, and Production', Proceedings. IEEE, pp. 320–329.

Ben-Ari, M., 2012. Mathematical logic for computer science. Springer Science & Business Media.

Bonneville, F., Perrard, C., Henrioud, J., 1995. A genetic algorithm to generate and evaluate assembly plans. In: Emerging Technologies and Factory Automation, 1995. ETFA'95, Proceedings., 1995 INRIA/IEEE Symposium on. Vol. 2. IEEE, pp. 231–239.

Brachman, R., Levesque, H., 2004. Knowledge Representation and Reasoning. Morgan Kaufmann.

Burbidge, J., 1993. Comment on clustering methods for finding gt groups and families. Journal of Manufacturing Systems 12 (5), 428–429.

Case, K., 1994. Using a design by features cad system for process capability modelling. Computer Integrated Manufacturing Systems 7 (1), 39–49.

Case, K., Wan Harun, W., 2000. Feature-based representation for manufacturing planning. International Journal of Production Research 38 (17), 4285–4300.

Catania, G., 1991. Form-features for mechanical design and manufacturing. Journal of engineering Design 2 (1), 21–43.

Chaube, A., Benyoucef, L., Tiwari, M., 2012. An adapted nsga-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system. Journal of Intelligent Manufacturing 23 (4), 1141–1155.

Chen, W., Tai, P., Deng, W., Hsieh, L., 2008. A three-stage integrated approach for assembly sequence planning using neural networks. Expert Systems with Applications 34 (3), 1777–1786.

Chen, X., Gao, S., Yang, Y., Zhang, S., 2012. Multi-level assembly model for top-down design of mechanical products. Computer-Aided Design 44 (10), 1033–1048.

Chung, D., Suh, S., 2008. Iso 14649-based nonlinear process planning implementation for complex machining. Computer-Aided Design 40 (5), 521–536.

Daaboul, J., Da Cunha, C., Bernard, A., Laroche, F., 2011. Design for mass customization: Product variety vs. process variety. CIRP Annals-Manufacturing Technology 60 (1), 169–174.

Danloy, J., Petit, F., Leroy, A., De Lit, P., Rekiek, B., 1999. A pragmatic approach for precedence graph generation. In: Assembly and Task Planning, 1999.(ISATP'99) Proceedings of the 1999 IEEE International Symposium on. IEEE, pp. 387–392.

De Lit, P., Delchambre, A., 2003. Integrated design of a product family and its assembly system. Springer Science & Business Media.

de Mello, L., Sanderson, A., 1991a. A basic algorithm for the generation of mechanical assembly sequences. In: Computer-Aided Mechanical Assembly Planning. Springer, pp. 163–190.

de Mello, L., Sanderson, A., 1991b. Representations for assembly sequences. In: Computer-aided mechanical assembly planning. Springer, pp. 129–162.

Delchambre, A., 1992. Computer aided Assembly Planning. Chapman & Hall, London. U.K.

Deneux, D., 1999. Introduction to assembly features: an illustrated synthesis methodology. Journal of Intelligent Manufacturing 10 (1), 29–39.

Denkena, B., Shpitalni, M., Kowalski, P., Molcho, G., Zipori, Y., 2007. Knowledge management in process planning. CIRP Annals-Manufacturing Technology 56 (1), 175–180.

Ding, L., Yue, Y., Ahmet, K., Jackson, M., Parkin, R., 2005. Global optimization of a feature-based process sequence using ga and ann techniques. International Journal of Production Research 43 (15), 3247–3272.

ElMaraghy, H., 1993. Evolution and future perspectives of capp. CIRP Annals-Manufacturing Technology 42 (2), 739–751.

ElMaraghy, H., 2006. Reconfigurable process plans for reconfigurable manufacturing. In: Proceedings of the 3rd International CIRP Sponsored Conference on Digital Enterprise Technology.

ElMaraghy, H., 2009. Changing and evolving products and systems–models and enablers. In: Changeable and reconfigurable manufacturing systems. Springer, pp. 25–45.

ElMaraghy, H., Schuh, G., ElMaraghy, W., Piller, F., Schönsleben, P., Tseng, M., Bernard, A., 2013. Product variety management. CIRP Annals-Manufacturing Technology 62 (2), 629–652.

ElMaraghy, H. A., 2007. Reconfigurable process plans for responsive manufacturing systems. In: Digital enterprise technology. Springer, pp. 35–44.

Eng, T., Ling, Z., Olson, W., McLean, C., 1999. Feature-based assembly modeling and sequence generation. Computers & Industrial Engineering 36 (1), 17–33.

Etienne, A., Dantan, J.-Y., Siadat, A., Martin, P., 2006. An improved approach for automatic process plan generation of complex borings. Computers in Industry 57 (7), 663–675.

Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J., 2014. Knowledge-based configuration: From research to business cases. Newnes.

Feng, S., 2003. A machining process planning activity model for systems integration. Journal of intelligent manufacturing 14 (6), 527–539.

Fenves, S., 2001. Core product model for representing design information. Tech. rep., National Institute of Standards and Technology.

Finger, S., Dixon, J., 1989. A review of research in mechanical engineering design. part i: Descriptive, prescriptive, and computer-based models of design processes. Research in engineering design 1 (1), 51–67.

Fujimoto, H., Ahmed, A., Iida, Y., Hanai, M., 2003. Assembly process design for managing manufacturing complexities because of product varieties. International Journal of Flexible Manufacturing Systems 15 (4), 283–307.

Garwood, D., 1988. Bills of material structured for excellence. Dogwood Publishing Co., Marietta, GA.

Givehchi, M.and Haghighi, A., Wang, L., 2015. Generic machining process sequencing through a revised enriched machining feature concept. Journal of Manufacturing Systems 37 (2), 564–575.

Gonzalez, F., Rosado, P., 2004. General information model for representing machining features in capp systems. International Journal of Production Research 42 (9), 1815–1842.

Goyal, K. K., Jain, P., Jain, M., 2013. A comprehensive approach to operation sequence similarity based part family formation in the reconfigurable manufacturing system. International Journal of Production Research 51 (6), 1762–1776.

Gu, P., 1994. A feature representation scheme for supporting integrated manufacturing. Computers & industrial engineering 26 (1), 55–71.

Gu, T., Xu, Z., Yang, Z., 2008. Symbolic obdd representations for mechanical assembly sequences. Computer-Aided Design 40 (4), 411–421.

Gujarathi, G., Ma, Y., 2011. Parametric cad/cae integration using a common data model. Journal of Manufacturing Systems 30 (3), 118–132.

Gupta, S., Krishnan, V., 1998. Product family-based assembly sequence design methodology. IIE transactions 30 (10), 933–945.

Hegge, H., Wortmann, J., 1991. Generic bill-of-material: a new product model. International Journal of Production Economics 23 (1), 117–128.

Henrioud, J., Bourjault, A., 1992. Computer aided assembly process planning. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 206 (1), 61–66.

Hong, G., Hu, L., Xue, D., Tu, Y., Xiong, Y., 2008. Identification of the optimal product configuration and parameters based on individual customer requirements on performance and costs in one-of-a-kind production. International Journal of Production Research 46 (12), 3297–3326.

Hsu, Y., Tai, P., Wang, M., Chen, W., 2011. A knowledge-based engineering system for assembly sequence planning. The International Journal of Advanced Manufacturing Technology 55 (5-8), 763–782.

Hu, C., Li, Z., Zheng, L., Li, N., Wen, P., 2008. An xml-based implementation of manufacturing route sheet documents for context-sensitive and web-based process planning. International Journal of Computer Integrated Manufacturing 21 (6), 647–656.

Hu, S. J., Ko, J., Weyand, L., ElMaraghy, H., Lien, T., Koren, Y., Bley, H., Chryssolouris, G., Nasr, N., Shpitalni, M., 2011. Assembly system design and operations for product variety. CIRP Annals-Manufacturing Technology 60 (2), 715–733.

Ji, W., Wang, L., Haghighi, A., Givehchi, M., Liu, X., 2016. A reachability based approach for machining feature sequencing. Journal of Manufacturing Systems 40, 96–104.

Jiao, J., Simpson, T., Siddique, Z., 2007. Product family design and platform-based product development: A state-of-the-art review. Journal of Intelligent Manufacturing 18 (1), 5–29.

Jiao, J., Tseng, M., Ma, Q., Zou, Y., 2000. Generic bill-of-materials-and-operations for high-variety production management. Concurrent Engineering 8 (4), 297–321.

Jiao, J., Zhang, L., Prasanna, K., 2004. Process variety modeling for process configuration in mass customization: An approach based on object-oriented petri nets with changeable structures. International Journal of Flexible Manufacturing Systems 16 (4), 335–361.

Jiménez, P., 2013. Survey on assembly sequencing: a combinatorial and geometrical perspective. Journal of Intelligent Manufacturing 24 (2), 235–250.

Kalpakjian, S., S.R., S., Vijay Sekar, K., 2009. Manufacturing engingeering and technology. Pearson.

Karger, D., 1993. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In: SODA. Vol. 93. pp. 21–30.

Kashkoush, M., ElMaraghy, H., 2015. Knowledge-based model for constructing master assembly sequence. Journal of Manufacturing Systems 34, 43–52.

Kotler, P., Armstrong, G., 2010. Principles of marketing. Pearson education.

Krause, F., Kimura, F., Kjellberg, T., Lu, S., Alting, L., Elmaraghy, H., Eversheim, W., Iwata, K., Suh, N., Tipnis, V., et al., 1993. Product modelling. CIRP Annals-Manufacturing Technology 42 (2), 695–706.

Krishnan, V., Ulrich, K., 2001. Product development decisions: A review of the literature. Management science 47 (1), 1–21.

Laakko, T., Mäntylä, M., 1994. Feature-based modelling of families of machined parts. In: ASME International Computers in Engineering Conference. Vol. 1. pp. 45–54.

Laperriere, L., ElMaraghy, H., 1996. Gapp: a generative assembly process planner. Journal of Manufacturing systems 15 (4), 282.

Lazzerini, B., Marcelloni, F., 2000. A genetic algorithm for generating optimal assembly plans. Artificial Intelligence in Engineering 14 (4), 319–329.

Lee, S., 1991. Backward assembly planning with dfa analysis. In: Computer-Aided Mechanical Assembly Planning. Springer, pp. 341–381.

Lee, S., 1992. Backward assembly planning with assembly cost analysis. In: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on. IEEE, pp. 2382–2391.

Li, B., Chen, L., Huang, Z., Zhong, Y., 2006a. Product configuration optimization using a multiobjective genetic algorithm. The International Journal of Advanced Manufacturing Technology 30 (1-2), 20–29.

Li, G., Zhou, L., An, L., Ji, J., Tan, C., Wang, Z., 2010. A system for supporting rapid assembly modeling of mechanical products via components with typical assembly features. The International Journal of Advanced Manufacturing Technology 46 (5-8), 785–800.

Li, S., Liu, Y., Wang, J., Zeng, H., 2014. An intelligent interactive approach for assembly process planning based on hierarchical classification of parts. The International Journal of Advanced Manufacturing Technology 70 (9-12), 1903–1914.

Li, W., Ong, S., Nee, A., 2002. Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts. International journal of production research 40 (8), 1899–1922.

Li, Y., Lu, Y., Liao, W., Lin, Z., 2006b. Representation and share of part feature information in web-based parts library. Expert Systems with Applications 31 (4), 697–704.

Liu, X., Yi, H., Ni, Z., 2013. Application of ant colony optimization algorithm in process planning optimization. Journal of Intelligent Manufacturing 24 (1), 1–13.

Liu, Z., Wang, L., 2007. Sequencing of interacting prismatic machining features for process planning. Computers in industry 58 (4), 295–303.

Ma, G., Zhang, Y., Nee, A., 2000. A simulated annealing-based optimization algorithm for process planning. International journal of production research 38 (12), 2671–2687.

Mäntylä, M., Sohlenius, G., 1993. Representation of process planning knowledge for part families. CIRP Annals-Manufacturing Technology 42 (1), 561–564.

Marian, R. M., Luong, L. H., Abhary, K., 2006. A genetic algorithm for the optimisation of assembly sequences. Computers & Industrial Engineering 50 (4), 503–527.

Markus, A., Váncza, J., Horvath, M., 1997. Process planning by retrieval and adaptation. Computers in industry 33 (1), 47–60.

Martinez, M., Favrel, J., Ghodous, P., 2000. Product family manufacturing plan generation and classification. Concurrent Engineering 8 (1), 12–23.

Martinez, M., Pham, V., Favrel, J., 2009. Optimal assembly plan generation: a simplifying approach. Journal of Intelligent Manufacturing 20 (1), 15–27.

Mascle, C., 2002. Feature-based assembly model for integration in computer-aided assembly. Robotics and Computer-Integrated Manufacturing 18 (5), 373–378.

McDermott, J., 1982. R1: A rule-based configurer of computer systems. Artificial intelligence 19 (1), 39–88.

Meyer, M., Utterback, J., 1993. The product family and the dynamics of core capability. Sloan management review 34 (3), 29.

Mittal, S., Falkenhainer, B., 1990. Dynamic constraint satisfaction. In: Proceedings Eighth National Conference on Artificial Intelligence. pp. 25–32.

Mokhtar, A., Xu, X., 2011. Machining precedence of 21/2d interacting features in a feature-based data model. Journal of Intelligent Manufacturing 22 (2), 145–161.

Mosemann, H., Rohrdanz, F., Wahl, F., 1997. Geometrical and physical cost evaluation for robot assembly sequence planning. In: Intelligent Engineering Systems, 1997. INES'97. Proceedings., 1997 IEEE International Conference on. IEEE, pp. 499–504.

Motavalli, S., Islam, A., 1997. Multi-criteria assembly sequencing. Computers & industrial engineering 32 (4), 743–751.

Nievergelt, J., 2000. Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In: International Conference on Current Trends in Theory and Practice of Computer Science. Springer, pp. 18–35.

Nilsson, N., 1980. Principles of artificial intelligence. Palo Alto, California: Tioga Press.

Niu, X., Ding, H., Xiong, Y., 2003. A hierarchical approach to generating precedence graphs for assembly planning. International Journal of Machine Tools and Manufacture 43 (14), 1473–1486.

Nonaka, Y., Erdős, G., Kis, T., Kovács, A., Monostori, L., Nakano, T., Váncza, J., 2013. Generating alternative process plans for complex parts. CIRP Annals-Manufacturing Technology 62 (1), 453–458.

Ong, N., Wong, Y., 1999. Automatic subassembly detection from a product model for disassembly sequence generation. The international journal of advanced manufacturing technology 15 (6), 425–431.

Pine, B., 1993. Mass Customization: The New Frontier in Business Competition. Harvard Business Press.

Pitiot, P., Aldanondo, M., Vareilles, E., 2014. Concurrent product configuration and process planning: Some optimization experimental results. Computers in Industry 65 (4), 610–621.

Qian, L., Gero, J., 1996. Function-behavior-structure paths and their role in analogy-based design. Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 10 (04), 289–312.

Qiao, L., Wang, X., Wang, S., 2000. A ga-based approach to machining operation sequencing for prismatic parts. International Journal of Production Research 38 (14), 3283–3303.

Requicha, A., Whalen, T., 1991. Representations for assemblies. In: Computer-aided mechanical assembly planning. Springer, pp. 15–39.

Russell, S., Norvig, P., Canny, J., Malik, J., Edwards, D., 2003. Artificial intelligence: a modern approach. Vol. 2. Prentice hall Upper Saddle River.

Salvador, F., Forza, C., 2004a. Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities. International journal of production economics 91 (3), 273–291.

Salvador, F., Forza, C., 2004b. Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities. International journal of production economics 91 (3), 273–291.

Scallan, P., 2003. Process planning: the design/manufacture interface. Butterworth-Heinemann.

Schierholt, K., 2001. Process configuration:combining the principles of product configuration and process planning. AI EDAM 15 (05), 411–414.

SCRA, June 2006. STEP application handbook. SCRA, 5300 International Boulevard, North Charleston, SC 29418, version 3 Edition.

Shah, J., 1991a. Assessment of features technology. Computer-Aided Design 23 (5), 331–343.

Shah, J., 1991b. Conceptual development of form features and feature modelers. Research in engineering design 2 (2), 93–108.

Shah, J., Rogers, M., 1993. Assembly modeling as an extension of feature-based design. Research in Engineering Design 5 (3-4), 218–237.

Shah, J. J., Yan, Y., Zhang, B., 1998. Dimension and tolerance modeling and transformations in feature based design and manufacturing. Journal of Intelligent Manufacturing 9 (5), 475–488.

Simpson, T. W., 1998. A concept exploration method for product family design. Ph.D. thesis, Georgia Institute of Technology.

Sormaz, D., 1994. A concept exploration method for product family design. Ph.D. thesis, University Of Southern California.

Sormaz, D. N., Khoshnevis, B., 2000. Modeling of manufacturing feature interactions for automated process planning. Journal of Manufacturing Systems 19 (1), 28.

Srinivasan, M., Sheng, P., 1999. Feature based process planning in environmentally conscious machining–part 2: macroplanning. Robotics and Computer-Integrated Manufacturing 15 (3), 271–281.

Stadzisz, P., Henrioud, J., 1995. Integrated design of product families and assembly systems. In: Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on. Vol. 2. IEEE, pp. 1290–1295.

Thomas, U., Barrenscheen, M., Wahl, F., 2003. Efficient assembly sequence planning using stereographical projections of c-space obstacles. In: Assembly and Task Planning, 2003. Proceedings of the IEEE International Symposium on. IEEE, pp. 96–102.

Tolio, T., Ceglarek, D., Elmaraghy, H., Fischer, A., Hu, S., Laperrirè, L., Newman, S., Váncza, J., 2010. Species-co-evolution of products, processes and production systems. CIRP Annals - Manufacturing Technology 59 (2), 672–693.

Tseng, H., Chang, C., Chang, S., 2005. Applying case-based reasoning for product configuration in mass customization environments. Expert Systems with Applications 29 (4), 913–925.

Ulrich, K., 1995. The role of product architecture in the manufacturing firm. Research policy 24 (3), 419–440.

Van Holland, W., Bronsvoort, W., 2000. Assembly features in modeling and planning. Robotics and computer-integrated manufacturing 16 (4), 277–294.

Wang, J., Liu, J., Zhong, Y., 2005. A novel ant colony algorithm for assembly sequence planning. The international journal of advanced manufacturing technology 25 (11-12), 1137–1143.

Wang, L., 2013. Machine availability monitoring and machining process planning towards cloud manufacturing. CIRP Journal of Manufacturing Science and Technology 6 (4), 263–273.

Wang, L., Cai, N., Feng, H., Liu, Z., 2006. Enriched machining feature-based reasoning for generic machining process sequencing. International Journal of Production Research 44 (8), 1479–1501.

Wang, L., Zhong, S., Zhang, Y., 2015. Process configuration based on generative constraint satisfaction problem. Journal of Intelligent Manufacturing, 1–13.

Wang, Y., Liu, J., 2010. Chaotic particle swarm optimization for assembly sequence planning. Robotics and Computer-Integrated Manufacturing 26 (2), 212–222.

Whitney, D., 2004. The datum flow chain. In: Mechanical assemblies: their design, manufacture, and role in product development. Springer, pp. 211–251.

Wolter, J., 1991. A combinatorial analysis of enumerative data structures for assembly planning. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. IEEE, pp. 611–618.

Xia, Q., Etienne, A., Dantan, J., Siadat, A., 2015. Product variety and reconfigurable process plan modeling for new manufacturing paradigms. In: 45th International Conference on Computers & Industrial Engineering 2015. Vol. 2. pp. 865–972.

Xia, Q., Etienne, A., Dantan, J., Siadat, A., 2016. Reconfigurable machining process planning for part variety in new manufacturing paradigms: Part i - definitions, models and applications(submitted paper).

Xie, H., Henderson, P., Kernahan, M., 2005. Modelling and solving engineering product configuration problems by constraint satisfaction. International Journal of Production Research 43 (20), 4455–4469.

Xu, X., Klemm, P., Proctor, F., Suh, S., 2006. Step-compliant process planning and manufacturing. International Journal of Computer Integrated Manufacturing 19 (6), 491–494.

Xu, X., Wang, L., Newman, S., 2011. Computer-aided process planning–a critical review of recent developments and future trends. International Journal of Computer Integrated Manufacturing 24 (1), 1–31.

Yang, D., Dong, M., Miao, R., 2008. Development of a product configuration system with an ontology-based approach. CAD Computer Aided Design 40 (8), 863–878.

Zhang, J., Xu, Z., Li, Y., Jiang, S., 2015. Framework for the integration of assembly modeling and simulation based on assembly feature pair. The International Journal of Advanced manufacturing Technology 78 (5-8), 765–780.

Zhang, L., 2014. Product configuration: A review of the state-of-the-art and future research. International Journal of Production Research 52 (21), 6381–6398.

Zhang, S., Gao, W., Merchant, M., et al., 1984. Tojicap:a system of computer aided process planning system for rotational parts. CIRP Annals-Manufacturing Technology 33 (1), 299–301.

Zhao, S., Li, Z., 2009. Formalized reasoning method for assembly sequences based on polychromatic sets theory. The International Journal of Advanced Manufacturing Technology 42 (9-10), 993–1004.

Zheng, L., Dong, H., Vichare, P., Nassehi, A., Newman, S., 2008. Systematic modeling and reusing of process knowledge for rapid process configuration. Robotics and Computer-Integrated Manufacturing 24 (6), 763–772.

Appendix A

Figures

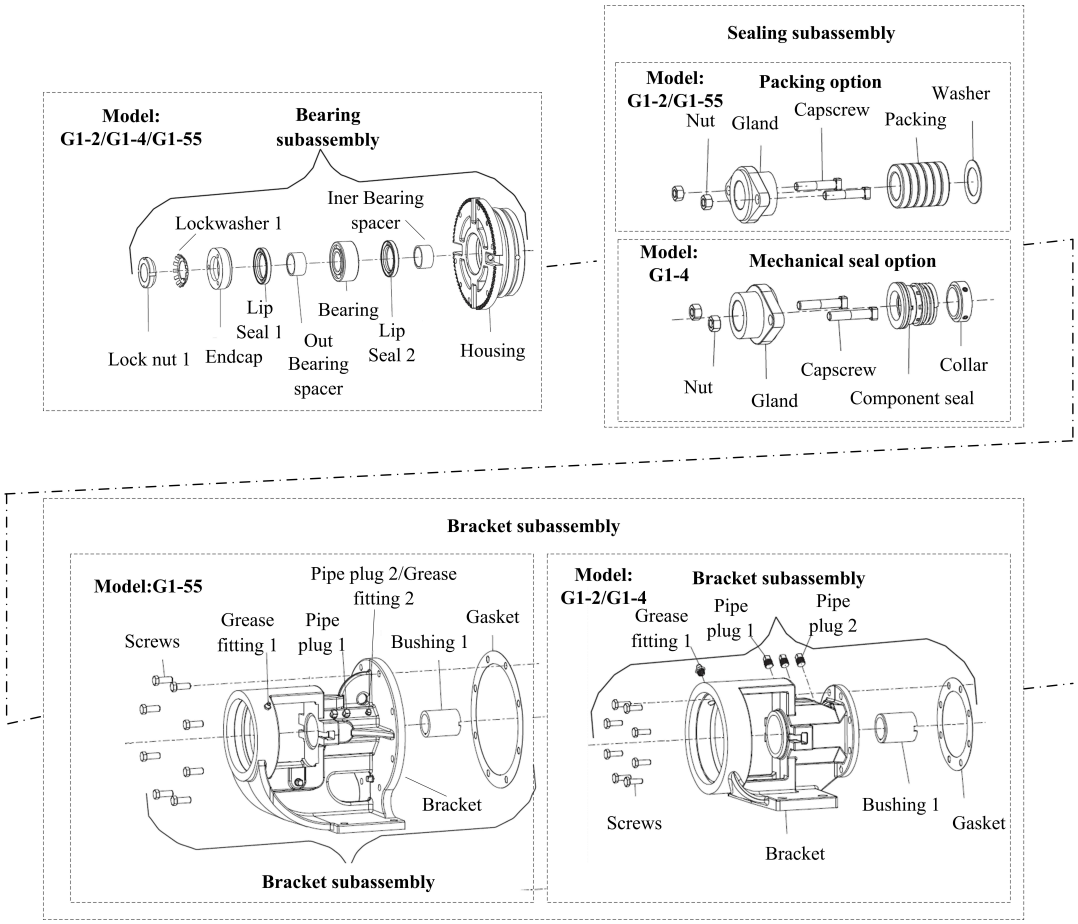


Figure A-1: Part structure of a gear pump family (Bearing subassembly/Sealing subassembly/Bracket subassembly)

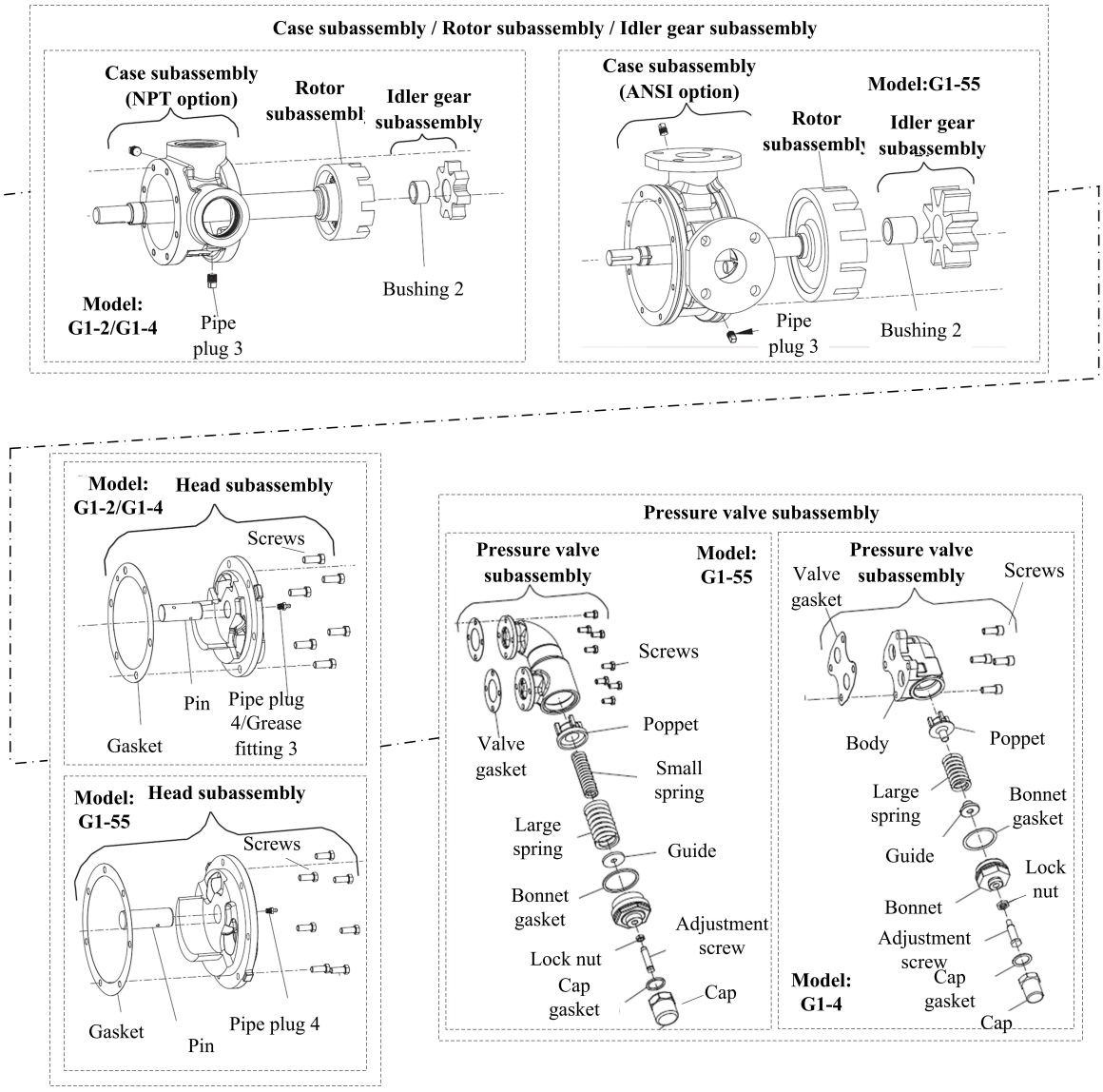


Figure A-2: Part structure of a gear pump family (Case assembly/Rotor subassembly/Idler gear subassembly/Head subassembly/Pressure valve subassembly)

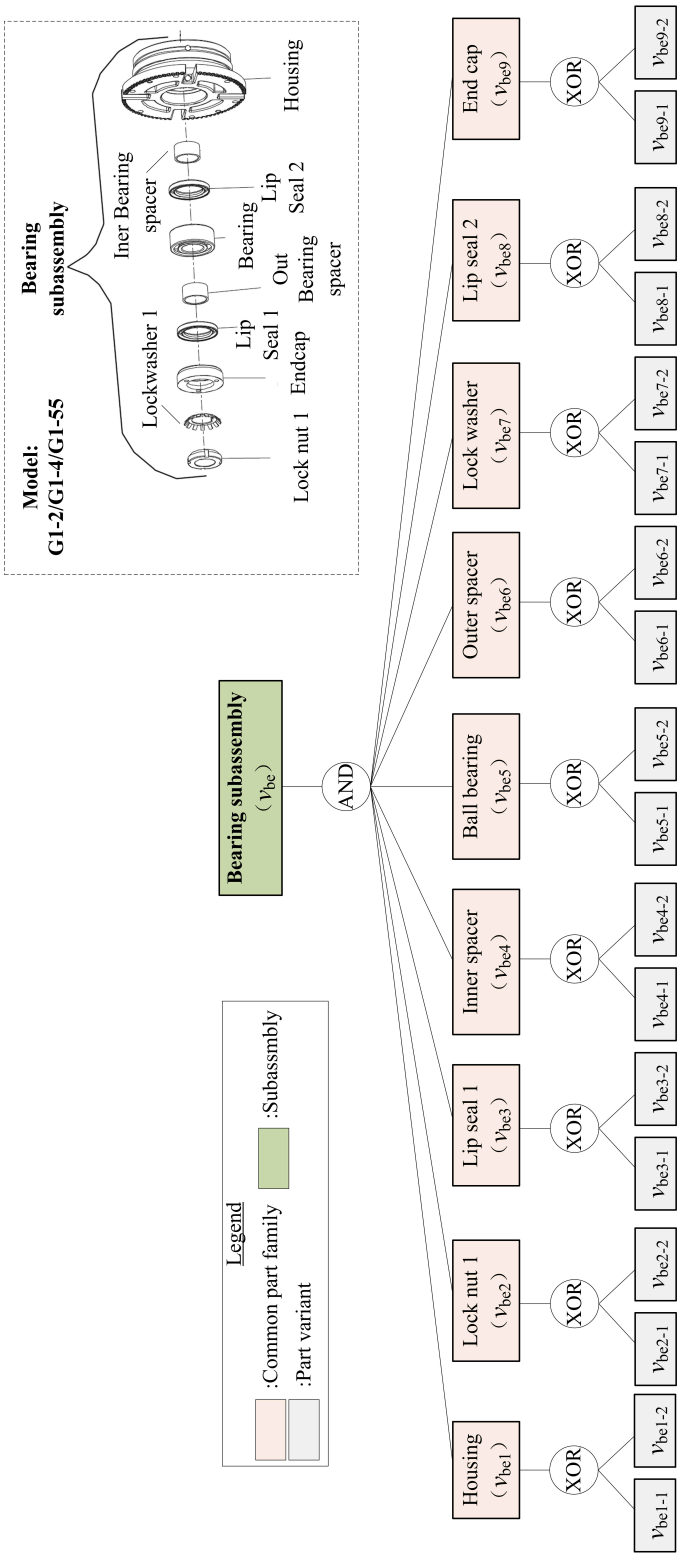


Figure A-3: Part structure of the bearing subassembly in the gear pump family

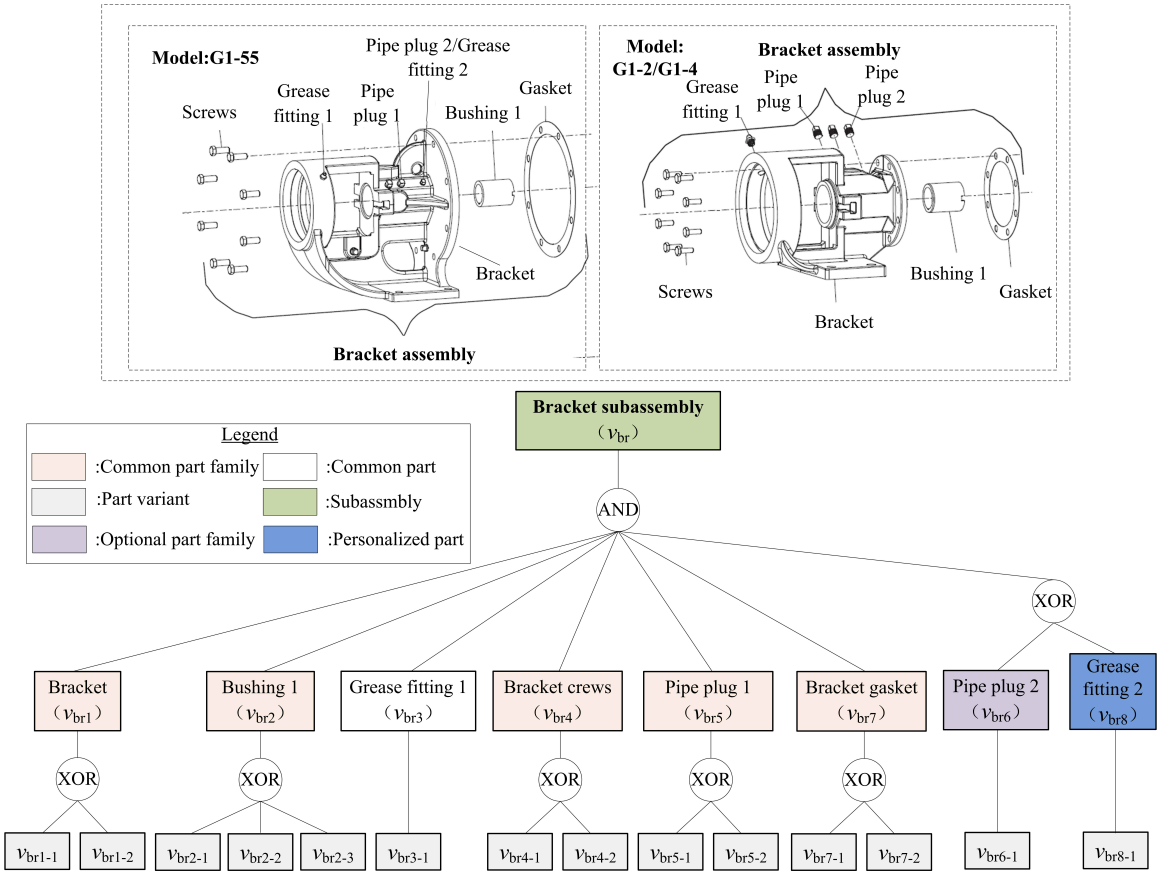


Figure A-4: Part structure of bracket subassembly in the gear pump family

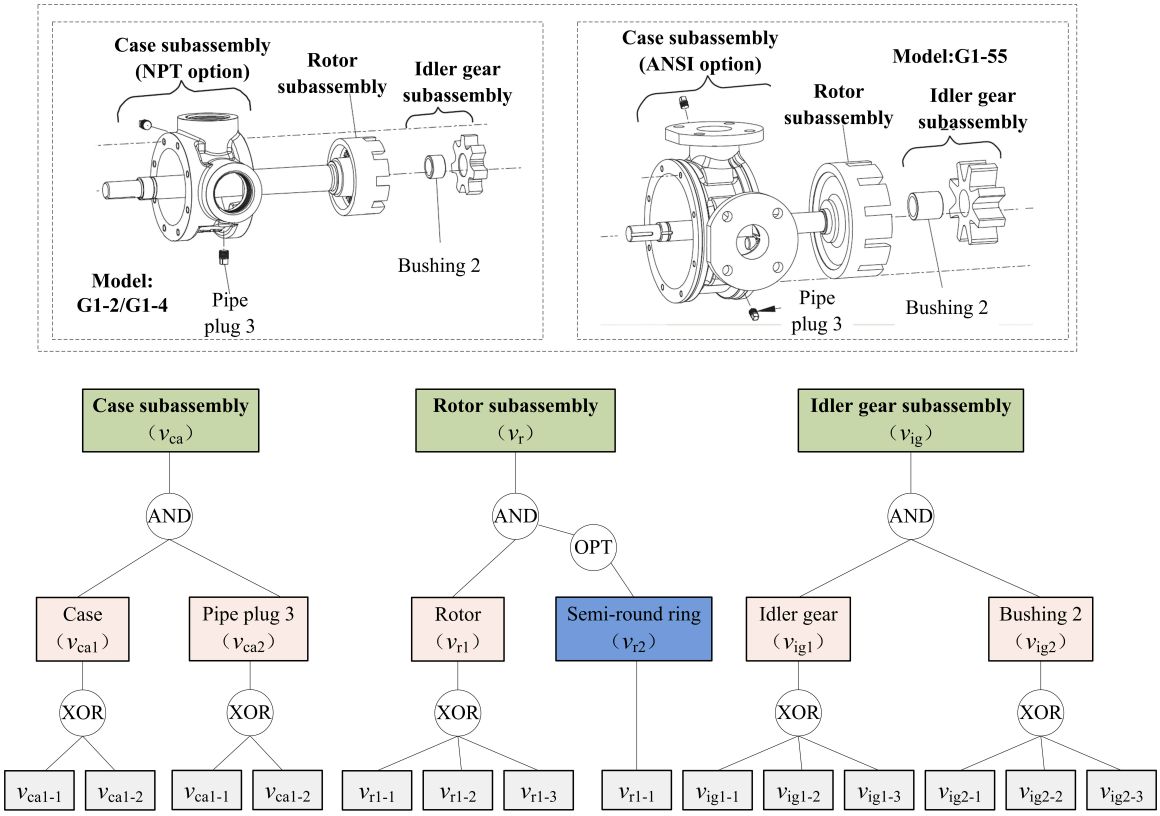


Figure A-5: Part structure of case/rotor/idler gear subassembly in the gear pump family

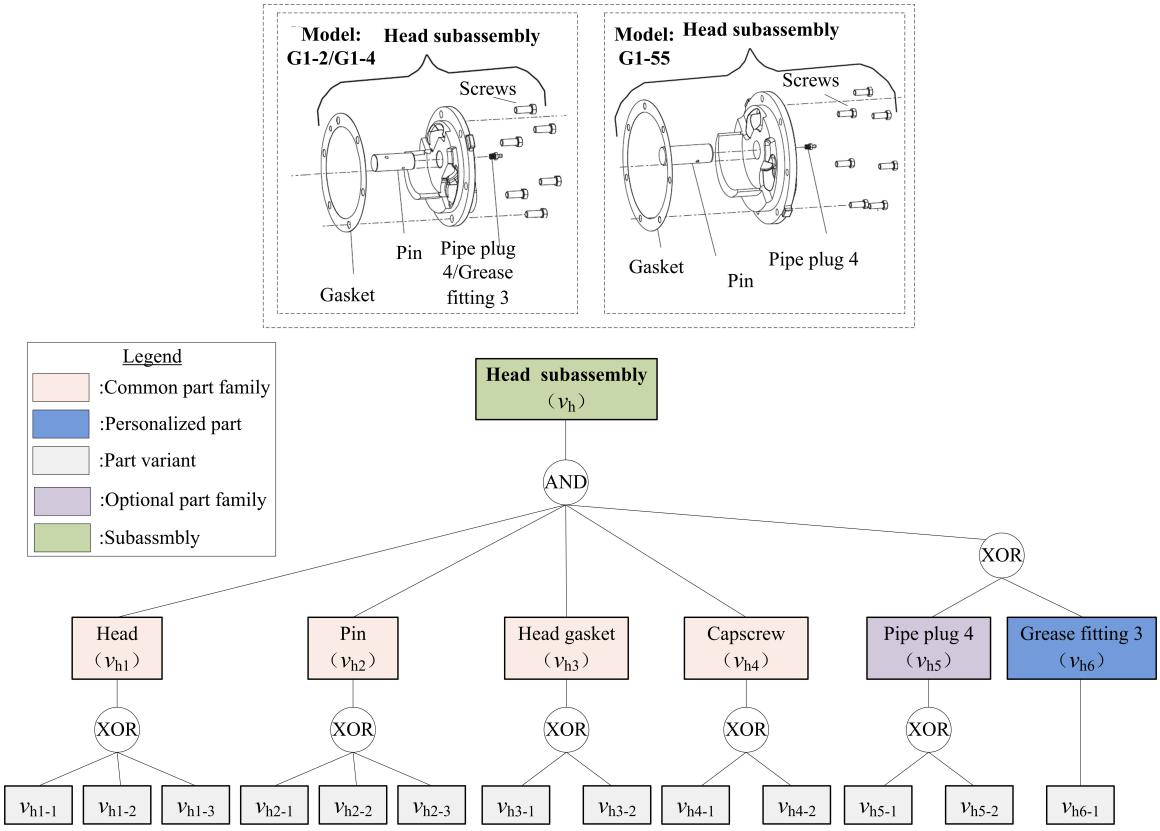


Figure A-6: Part structure of the head subassembly in the gear pump family

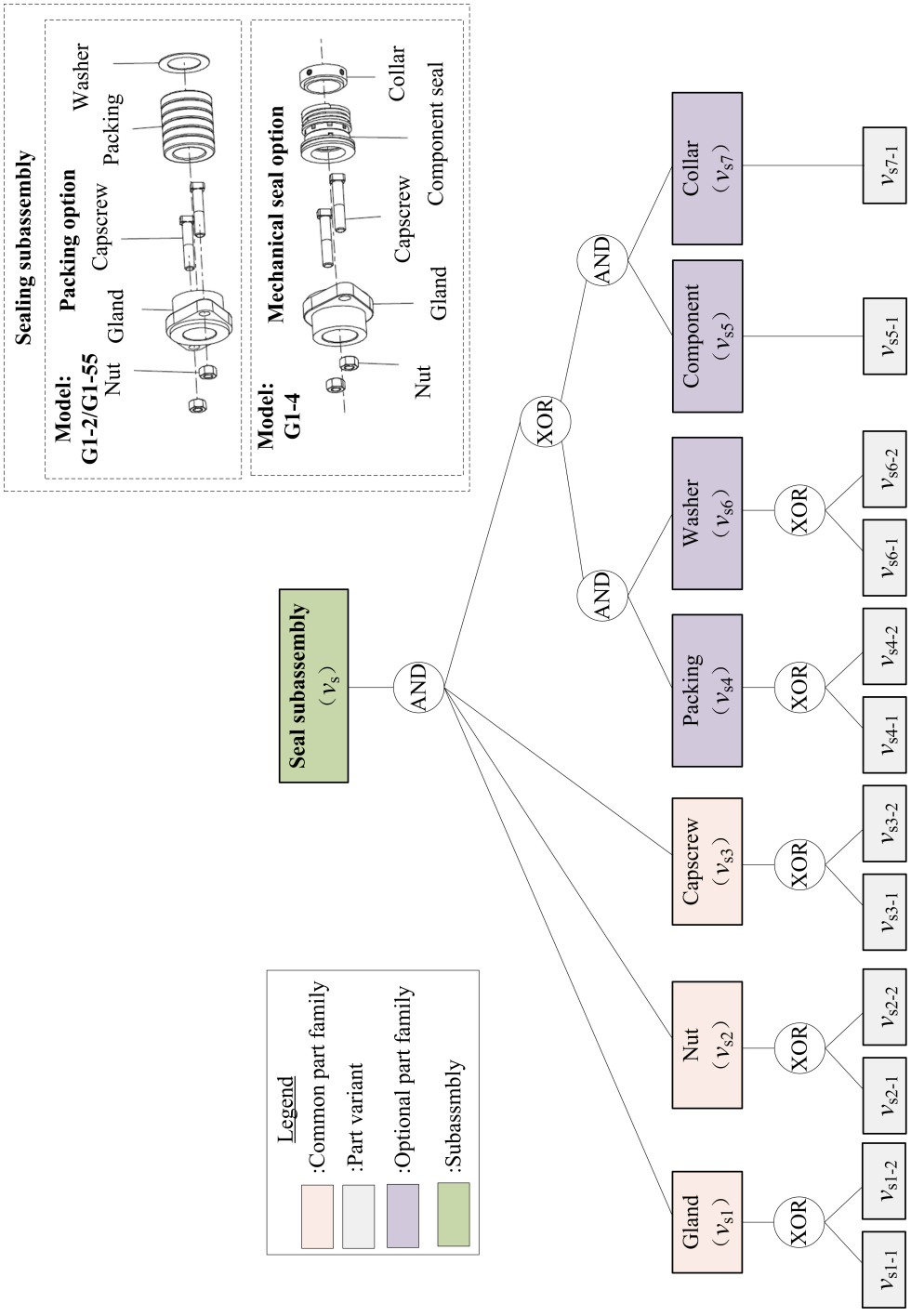


Figure A-7: Part structure of the sealing subassembly in the gear pump family

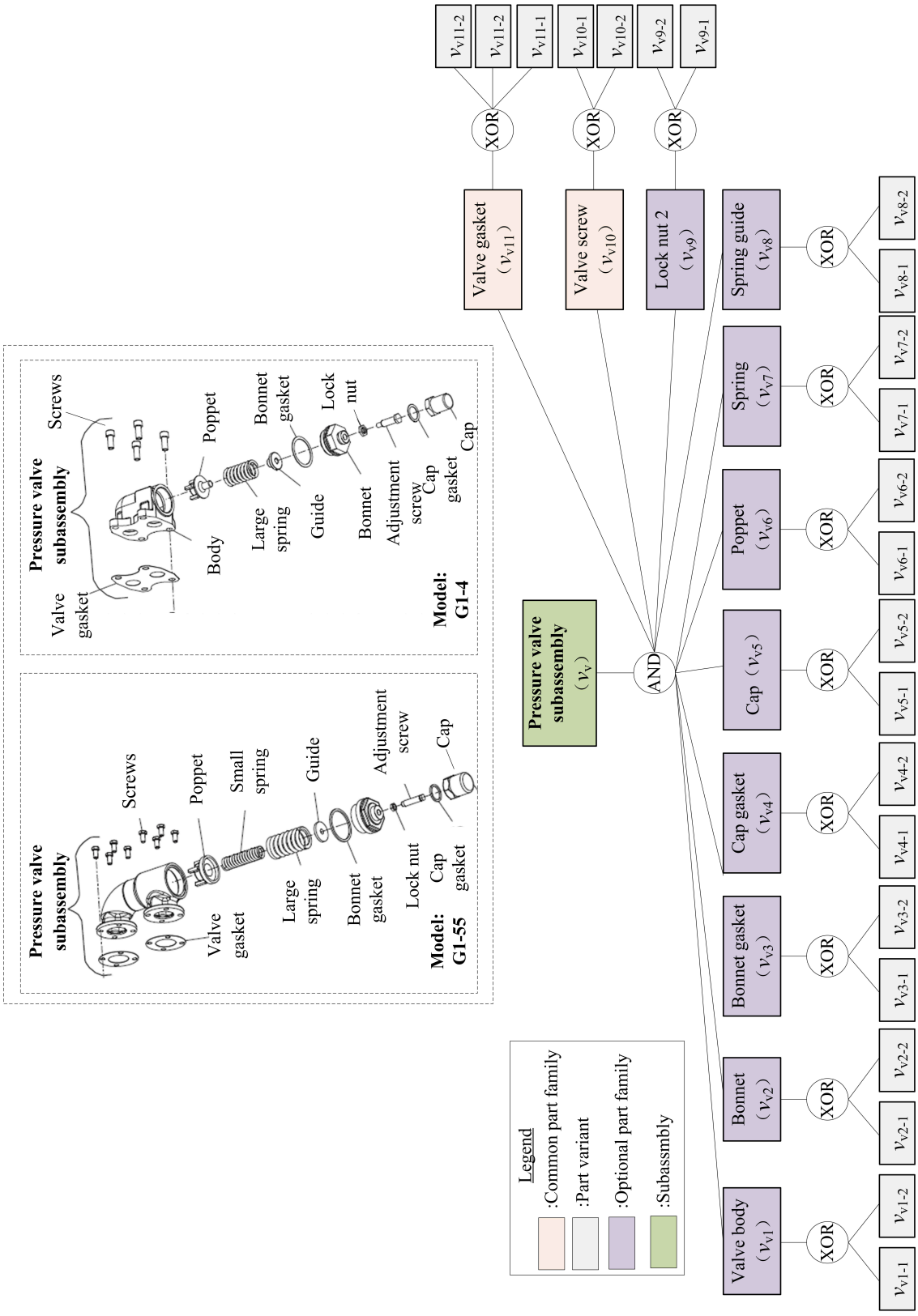


Figure A-8: Part structure of the pressure relief valve subassembly in the gear pump family

Appendix B

Tables

Table B.1: Part list of the three gear pump variants

Subassemblies	Part families/ common parts	Part variants		
		Variant 1	Variant 2	Variant 3
Bearing subassembly (v_{be})	Housing (v_{be1})	$v_{\mathrm{be1\text{-}1}}$		$v_{\mathrm{be1\text{-}2}}$
	Lock nut 1 (v_{be2})	$v_{\mathrm{be2\text{-}1}}$		$v_{\mathrm{be2\text{-}2}}$
	Lip seal 1 (v_{be3})	$v_{\mathrm{be3\text{-}1}}$		$v_{\mathrm{be3\text{-}2}}$
	Inner bearing spacer (v_{be4})	$v_{\mathrm{be4\text{-}1}}$		$v_{\mathrm{be4\text{-}2}}$
	Ball bearing (v_{be5})	$v_{\mathrm{be5\text{-}1}}$		$v_{\mathrm{be5\text{-}2}}$
	Outer bearing spacer (v_{be6})	$v_{\mathrm{be6\text{-}1}}$		$v_{\mathrm{be6\text{-}2}}$
	Lock washer (v_{be7})	$v_{\mathrm{be7\text{-}1}}$		$v_{\mathrm{be7\text{-}2}}$
	Lip seal 2 (v_{be8})	$v_{\mathrm{be8\text{-}1}}$		$v_{\mathrm{be8\text{-}2}}$
	End cap (v_{be9})	$v_{\mathrm{be9\text{-}1}}$		$v_{\mathrm{be9\text{-}2}}$
Bracket subassembly (v_{br})	Bracket (v_{br1})	$v_{\mathrm{br1\text{-}1}}$		$v_{\mathrm{br1\text{-}2}}$
	Bushing 1 (v_{br2})	$v_{\mathrm{br2\text{-}1}}$	$v_{\mathrm{br2\text{-}2}}$	$v_{\mathrm{br2\text{-}3}}$
	Grease fitting 1 (v_{br3})		$v_{\mathrm{br3\text{-}1}}$	
	Bracket screws (v_{br4})	$v_{\mathrm{br4\text{-}1}}$		$v_{\mathrm{br4\text{-}2}}$
	Pipe plug 1 (v_{br5})	$v_{\mathrm{br5\text{-}1}}$		$v_{\mathrm{br5\text{-}2}}$
	Pipe plug 2 (v_{br6})	-	$v_{\mathrm{br6\text{-}1}}$	
	Bracket gasket (v_{br7})	$v_{\mathrm{br7\text{-}1}}$		$v_{\mathrm{br7\text{-}2}}$
	Grease fitting 2 (v_{br8})	$v_{\mathrm{br8\text{-}1}}$	-	-
Case subassembly (v_{ca})	Case (v_{ca1})	$v_{\mathrm{ca1\text{-}1}}$		$v_{\mathrm{ca1\text{-}2}}$
	Pipe plug 3 (v_{ca2})	$v_{\mathrm{ca2\text{-}1}}$		$v_{\mathrm{ca2\text{-}2}}$
Rotor subassembly (v_{r})	Rotor (v_{r1})	$v_{\mathrm{r1\text{-}1}}$	$v_{\mathrm{r1\text{-}2}}$	$v_{\mathrm{r1\text{-}3}}$
	Semi-round rings (v_{r2})	-		$v_{\mathrm{r2\text{-}1}}$
Idler gear subassembly (v_{ig})	Idler gear (v_{ig1})	$v_{\mathrm{ig1\text{-}1}}$	$v_{\mathrm{ig1\text{-}2}}$	$v_{\mathrm{ig1\text{-}3}}$
	Bushing 2 (v_{ig2})	$v_{\mathrm{ig2\text{-}1}}$	$v_{\mathrm{ig2\text{-}2}}$	$v_{\mathrm{ig2\text{-}3}}$
Head subassembly (v_{h})	Head (v_{h1})	$v_{\mathrm{h1\text{-}1}}$	$v_{\mathrm{h1\text{-}2}}$	$v_{\mathrm{h1\text{-}3}}$
	Pin (v_{h2})	$v_{\mathrm{h2\text{-}1}}$	$v_{\mathrm{h2\text{-}2}}$	$v_{\mathrm{h2\text{-}3}}$
	Head gasket (v_{h3})	$v_{\mathrm{h3\text{-}1}}$		$v_{\mathrm{h3\text{-}2}}$

Table B.1: Part list of the three gear pump variants(continued)

Subassemblies	Part families/ common parts	Part variants		
		Variant 1	Variant 2	Variant 3
Head subassembly (v_{h})	Capscrew ($v_{\mathrm{h}4}$)	$v_{\mathrm{h}4\text{-}1}$		$v_{\mathrm{h}4\text{-}2}$
	Pipe plug 4 ($v_{\mathrm{h}5}$)	-	$v_{\mathrm{h}5\text{-}1}$	
	Grease fitting 3 ($v_{\mathrm{h}6}$)	$v_{\mathrm{h}6\text{-}1}$	-	
Seal subassembly (v_{s})	Gland ($v_{\mathrm{s}1}$)	$v_{\mathrm{s}1\text{-}1}$		$v_{\mathrm{s}1\text{-}2}$
	Nut ($v_{\mathrm{s}2}$)	$v_{\mathrm{s}2\text{-}1}$		$v_{\mathrm{s}2\text{-}2}$
	Capscrew ($v_{\mathrm{s}3}$)	$v_{\mathrm{s}3\text{-}1}$		$v_{\mathrm{s}3\text{-}2}$
	Packing ($v_{\mathrm{s}4}$)	$v_{\mathrm{s}4\text{-}1}$	-	$v_{\mathrm{s}4\text{-}2}$
	Component ($v_{\mathrm{s}5}$)	-	$v_{\mathrm{s}5\text{-}1}$	-
	Washer ($v_{\mathrm{s}6}$)	$v_{\mathrm{s}6\text{-}1}$	-	$v_{\mathrm{s}6\text{-}2}$
	Collar ($v_{\mathrm{s}7}$)	-	$v_{\mathrm{s}7\text{-}1}$	-
Pressure relief valve subassembly (v_{v})	Valve body ($v_{\mathrm{v}1}$)	-	$v_{\mathrm{v}1\text{-}1}$	$v_{\mathrm{v}1\text{-}2}$
	Bonnet ($v_{\mathrm{v}2}$)	-	$v_{\mathrm{v}2\text{-}1}$	$v_{\mathrm{v}2\text{-}2}$
	Bonnet gasket (v_3)	-	$v_{\mathrm{v}3\text{-}1}$	$v_{\mathrm{v}3\text{-}2}$
	Cap gasket ($v_{\mathrm{v}4}$)	-	$v_{\mathrm{v}4\text{-}1}$	$v_{\mathrm{v}4\text{-}2}$
	Cap ($v_{\mathrm{v}5}$)	-	$v_{\mathrm{v}5\text{-}1}$	$v_{\mathrm{v}5\text{-}2}$
	Poppet ($v_{\mathrm{v}6}$)	-	$v_{\mathrm{v}6\text{-}1}$	$v_{\mathrm{v}6\text{-}2}$
	Spring ($v_{\mathrm{v}7}$)	-	$v_{\mathrm{v}7\text{-}1}$	$v_{\mathrm{v}7\text{-}2}$
	Spring guide($v_{\mathrm{v}8}$)	-	$v_{\mathrm{v}8\text{-}1}$	$v_{\mathrm{v}8\text{-}2}$
	Lock nut 2 ($v_{\mathrm{v}9}$)	-	$v_{\mathrm{v}9\text{-}1}$	$v_{\mathrm{v}9\text{-}2}$
	Valve screw ($v_{\mathrm{v}10}$)	-	$v_{\mathrm{v}10\text{-}1}$	$v_{\mathrm{v}10\text{-}2}$
	Valve gasket ($v_{\mathrm{v}11}$)	$v_{\mathrm{v}11\text{-}1}$	$v_{\mathrm{v}11\text{-}2}$	$v_{\mathrm{v}11\text{-}3}$
Cover subassembly (v_{co})	Cover ($v_{\mathrm{co}1}$)	$v_{\mathrm{co}1\text{-}1}$	-	-

Table B.2: Variables in the DCSP for the oil pump body family

Variables		Variety components
v_{f}	v_{f1}	Positioning the driving gear (F1)
	v_{f2}	Positioning the driving shaft (F2)
	v_{f3}	Positioning the oil outlet (F3)
	v_{f4}	Positioning the pressure valve (F4)
v_{fc}	v_{pc1}	Pocket cluster for F1 (PC1)
	v_{cc1}	Chamfer cluster for F1 (CC1)
	v_{hc2}	Hole cluster for F2 (HC2)
	v_{pc2}	Pocket cluster for F2 (PC2)
	v_{hc3}	Hole cluster for F3 (HC3)
	v_{pc4}	Pocket cluster for F4 (PC4)
	v_{cc4}	Chamfer cluster for F4 (CC4)
v_{fv}	$v_{\mathrm{pc1\text{-}1}}$	Pocket variant in PC1 (PO100)
	$v_{\mathrm{pc1\text{-}2}}$	Pocket variant in PC2 (PO200)
	$v_{\mathrm{cc1\text{-}1}}$	Chamfer variant in CC1 (CH100)
	$v_{\mathrm{cc1\text{-}2}}$	Chamfer variant in CC1 (CH200)
	$v_{\mathrm{hc2\text{-}1}}$	Hole variant in HC2 (CY110)
	$v_{\mathrm{hc2\text{-}2}}$	Hole variant in HC2 (CY210)
	$v_{\mathrm{pc2\text{-}1}}$	Pocket variant in PC2 (PO210)
	$v_{\mathrm{hc3\text{-}1}}$	Hole variant in HC3 (CY120)
	$v_{\mathrm{pc4\text{-}1}}$	Pocket variant in PC4 (PO230)
	$v_{\mathrm{pc4\text{-}2}}$	Pocket variant in PC4 (PO231)
	$v_{\mathrm{cc4\text{-}1}}$	Chamfer variant in CC4 (CH230)
v_{rm}	v_{rm1}	RMOP for PC1
	v_{rm2}	RMOP for CC1
	v_{rm3}	RMOP for PC2
	v_{rm4}	RMOP for HC2
	v_{rm5}	RMOP for HC3
	v_{rm6}	RMOP for PC4
	v_{rm7}	RMOP for CC4
v_{pr}	v_{pr1}	HC2 $solidBefore$ PC1
	v_{pr2}	PC1 $hardBefore$ CC1
	v_{pr3}	HC2 $hardBefore$ HC3
	v_{pr4}	HC2 $hardBefore$ PC4
	v_{pr5}	PC4 $hardBefore$ CC4
	v_{pr6}	PO231 $softImmeBefore$ PO230

Appendix C

Codes for the proposed algorithms

Code for integrated part configuration and RMPP configuration

```
:- lib(ic).
:- lib(ech).
:- local struct(fms(f1, f2, f3, f4)).
:- local struct(fcs(pc1, cc1, hc2, pc2, hc3, pc4, cc4)).
:- local struct(part_family(fms, fcs, fvs, rms, prs)).
:- local struct(rms(rm1, rm2, rm3, rm4, rm5, rm6, rm7)).
:- local struct(fvs(pc1_1, pc1_2, cc1_1, cc1_2, hc2_1, hc2_2, pc2_1, hc3_1,
    pc4_1, pc4_2, cc4_1)).
:- local struct(prs(pr1, pr2, pr3, pr4, pr5, pr6)).
:- constraints cc1/3, cc2/4, cc3/4, cc4/4, cc11/3, cc12/4.

cc1(A:L, B:M, Active)<=>memberd(A:L, Active),memberd(B:M, Active)|L $= M.
cc11(A:L, B:M, Active)<=>memberd(A:L, Active)|L $= M.
cc11(A:L, B:M, Active)<=>not memberd(A:L, Active)|M = 0.
cc12(A:L, B:M, C:N, Active)<=>memberd(A:L, Active),memberd(B:M, Active)|N $= (L
    and M).
cc12(A:L, B:M, C:N, Active)<=>not memberd(A:L, Active)|N = 0.
cc12(A:L, B:M, C:N,Active)<=>not memberd(B:M, Active)|N = 0.
cc2(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active),
    memberd(C:N, Active)|L $= (M and N).
```

```
cc3(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active),
    memberd(C:N, Active)|L $= ((M and N) or (neg M and N)).
cc4(A:L, B:M, C:N, Active)<=>memberd(A:L, Active), memberd(B:M, Active),
    memberd(C:N, Active)|L $= ((M and neg N) or (neg M and N)).

solve_dcsp(part_family(fms(f1:F1,f2:F2,f3:F3,f4:F4),
           fcs(pc1:PC1,cc1:CC1,hc2:HC2,pc2:PC2,hc3:HC3,pc4:PC4,cc4:CC4),
           fvs(pc1_1:PC1_1, pc1_2:PC1_2, cc1_1:CC1_1, cc1_2:CC1_2, hc2_1:HC2_1,
               hc2_2:HC2_2, pc2_1:PC2_1, hc3_1:HC3_1, pc4_1:PC4_1, pc4_2:PC4_2,
               cc4_1:CC4_1),
           rms(rm1:RM1,rm2:RM2,rm3:RM3,rm4:RM4,rm5:RM5,rm6:RM6,rm7:RM7),
           prs(pr1:PR1,pr2:PR2,pr3:PR3,pr4:PR4,pr5:PR5,pr6:PR6))):-

[F1,F2,F3,F4]::[0,1],
[PC1,CC1,HC2,PC2,HC3,PC4,CC4]::[0,1],
[PC1_1,PC1_2,CC1_1,CC1_2,HC2_1,HC2_2,PC2_1,HC3_1,PC4_1,PC4_2,CC4_1]::[0,1],
[RM1,RM2,RM3,RM4,RM5,RM6,RM7]::[0,1],
[PR1,PR2,PR3,PR4,PR5,PR6]::[0,1],
Active = [f1:F1,f2:F2,f3:F3,f4:F4],

((F1 and F2 and F3 and neg F4) or (F1 and F2 and neg F3 and F4)) $=1,
find_solution(Active),
activate(f1:F1,[pc1:PC1,cc1:CC1],Active,New_active1),
activate(f2:F2,[pc2:PC2,hc2:HC2],New_active1,New_active2),
activate(f3:F3,[hc3:HC3],New_active2,New_active3),
activate(f4:F4,[pc4:PC4,cc4:CC4],New_active3,New_active4),
cc2(f1:F1,pc1:PC1,cc1:CC1,New_active4),
cc3(f2:F2,pc2:PC2,hc2:HC2,New_active4),
cc2(f4:F4,pc4:PC4,cc4:CC4,New_active4),
cc1(f3:F3,hc3:HC3,New_active4),

find_solution(New_active4),

activate(pc1:PC1,[pc1_1:PC1_1,pc1_2:PC1_2],New_active4,New_active5),
activate(cc1:CC1,[cc1_1:CC1_1,cc1_2:CC1_2],New_active5,New_active6),
activate(pc2:PC2,[pc2_1:PC2_1],New_active6,New_active7),
```

```
activate(hc2:HC2,[hc2_1:HC2_1,hc2_2:HC2_2],New_active7,New_active8),
activate(hc3:HC3,[hc3_1:HC3_1],New_active8,New_active9),
activate(pc4:PC4,[pc4_1:PC4_1,pc4_2:PC4_2],New_active9,New_active10),
activate(cc4:CC4,[cc4_1:CC4_1],New_active10,New_active11),
cc4(pc1:PC1,pc1_1:PC1_1,pc1_2:PC1_2,New_active11),
cc4(cc1:CC1,cc1_1:CC1_1,cc1_2:CC1_2,New_active11),
cc1(pc2:PC2,pc2_1:PC2_1,New_active11),
cc4(hc2:HC2,hc2_1:HC2_1,hc2_2:HC2_2,New_active11),
cc1(hc3:HC3,hc3_1:HC3_1,New_active11),
cc2(pc4:PC4,pc4_1:PC4_1,pc4_2:PC4_2,New_active11),
cc1(cc4:CC4,cc4_1:CC4_1,New_active11),
cc1(pc1_1:PC1_1,cc1_1:CC1_1,New_active11),
cc1(pc1_2:PC1_2,pc2_1:PC2_1,New_active11),
cc1(pc1_1:PC1_1,hc3_1:HC3_1,New_active11),
cc1(pc1_1:PC1_1,hc2_1:HC2_1,New_active11),
cc1(pc1_2:PC1_2,f4:F4,New_active11),
cc1(pc1_2:PC1_2,pc2:PC2,New_active11),

find_solution(New_active11),

cc11(pc1:PC1,rm1:RM1,New_active11),
cc11(cc1:CC1,rm2:RM2,New_active11),
cc11(pc2:PC2,rm3:RM3,New_active11),
cc11(hc2:HC2,rm4:RM4,New_active11),
cc11(hc3:HC3,rm5:RM5,New_active11),
cc11(pc4:PC4,rm6:RM6,New_active11),
cc11(cc4:CC4,rm7:RM7,New_active11),
cc12(hc2:HC2,pc1:PC1,pr1:PR1,New_active11),
cc12(pc1:PC1,cc1:CC1,pr2:PR2,New_active11),
cc12(hc2:HC2,hc3:HC3,pr3:PR3,New_active11),
cc12(hc2:HC2,pc4:PC4,pr4:PR4,New_active11),
cc12(pc4:PC4,cc4:CC4,pr5:PR5,New_active11),
cc11(pc4:PC2,pr6:PR6,New_active11),

find_solution([rm1:RM1, rm2:RM2, rm3:RM3, rm4:RM4, rm5:RM5, rm6:RM6, rm7:RM7,
    pr1:PR1, pr2:PR2, pr3:PR3, pr4:PR4, pr5:PR5, pr6:PR6]),
```

```
print_configuration(part_family(fms(f1:F1, f2:F2, f3:F3, f4:F4),
                        fcs(pc1:PC1, cc1:CC1, hc2:HC2, pc2:PC2, hc3:HC3, pc4:PC4,
                            cc4:CC4),
                        fvs(pc1_1:PC1_1, pc1_2:PC1_2, cc1_1:CC1_1, cc1_2:CC1_2,
                            hc2_1:HC2_1, hc2_2:HC2_2, pc2_1:PC2_1, hc3_1:HC3_1,
                            pc4_1:PC4_1, pc4_2:PC4_2, cc4_1:CC4_1),
                        rms(rm1:RM1, rm2:RM2, rm3:RM3, rm4:RM4, rm5:RM5, rm6:RM6,
                            rm7:RM7),
                        prs(pr1:PR1, pr2:PR2, pr3:PR3, pr4:PR4, pr5:PR5,
                            pr6:PR6))).

find_solution([]):- true.
find_solution([_:D|Active1]):-
   indomain(D),
   find_solution(Active1).

memberd(_,[]):-fail.
memberd(X:_,[H:_|_]):-
   X = H,!.
memberd(M:_,[_:_|Ts]):-
   memberd(M:_,Ts),!.

activate(_:Head,L2,Active,New_active):-
   ((nonvar(Head),Head =1) -> add(L2,Active,New_active); New_active = Active).

add([],List2,New_list):- New_list=List2.
add([E1|Es],List2,New_list):-
   add(Es,[E1|List2],New_list).

print_configuration(part_family(Fms,Fcs,Fvs,Rms,Prs)):-

   writeln("Find the following configuration result:"),
   write("     Function Modules:"),
   (foreacharg(Fm,Fms) do write_configuration(Fm)),
   nl,
```

```
   write("     Feature Clusters:"),
   (foreacharg(Fc,Fcs) do write_configuration(Fc)),
   nl,
   write("     Feature Variants:"),
   (foreacharg(Fv,Fvs) do write_configuration(Fv)),
   nl,
   write("     RMOPs:"),
   (foreacharg(Rm,Rms) do write_configuration(Rm)),
   nl,
   write("     Precedence relations:"),
   (foreacharg(Pr,Prs) do write_configuration(Pr)),
   nl.

write_configuration(A:B):-
   B==1-> write(A),write(" ");true.
```

Prolog code for the depth-first RMOP generation algorithm

```
:- local struct(capabilities(dt,sf,tp,cy)).
:- local struct(process(type,capabilities:capabilities)).
:- local struct(type(tdrilling,nboring,reaming,pboring)).
:- local struct(demands(dt,sf,tp,cy)).

solve(List,Demand,Results):-
   member(Start,List),
   first_process_constraint(Start),
   depthfirst([Start],Demand,List,Results1),
   reverse(Results1,Results).

depthfirst(Path,Demand,_,Path):-
   goal(Path,Demand).

depthfirst(Path,Demand,List,Sol):-
   s(Path,Node1,List),
   depthfirst([Node1|Path],Demand,List,Sol).

first_process_constraint(N):-
   N \= 2,
   N \= 3,
   N \= 4,
   N \= 5.

goal(Processes,Demand):-
   get_capabilities(Processes,Capabilities),
   satisfy(Demand,Capabilities).

get_capabilities([Node1|Nodes],Capabilities):-
   get_process(Node1,Process1),
   initial_capabilities(Process1,Capabilities).
```

```
initial_capabilities(process{dt:DT1, sf:SF1, tp:TP1, cy:CY1},
    capabilities{dt:DT2, sf:SF2, tp:TP2, cy:CY2}):-
   DT2 is DT1,
   SF2 is SF1,
   TP2 is TP1,
   CY2 is CY1.

get_process(M,process{type:Type1,capabilities:Cap1}):-
   Type1 is M,
   type_capabilities_constraint(Type1,Cap1).

type_capabilities_constraint(Type1, Cap1) :-
   relates(Type1, [tdrilling of type, nboring of type,reaming of type,pboring of
       type],
        Cap1, [capabilities{dt:9,sf:1.6,tp:0.05,cy:0.1},
              capabilities{dt:6,sf:0.2,tp:0.03,cy:0.01},
              capabilities{dt:7,sf:0.2,tp:0.05,cy:0.01},
              capabilities{dt:6,sf:0.2,tp:0.003,cy:0.001}]).
relates(X,Xs,Y,Ys) :-
   element(I,Xs,X),
   nth1(I,Ys,Y).

satisfy(demands{dt:DT, sf:SF, tp:TP, cy:CY}, capabilities{dt:CDT, sf:CSF, tp:CTP,
    cy:CCY}):-
   CDT =< DT,
   CSF =< SF,
   CTP =< TP,
   CCY =< CY.

s([Node1|Path],Node,List):-
   member(Node,List),
   not member(Node,[Node1|Path]),
   get_capabilities([Node1|Path],Capabilities),
   capability_compatible(Capabilities,Node),
   operation_compatible(Node1,Node).
```

```
capability_compatible(Capabilities,Node):-
   get_process(Node,Process),
   compatible(Capabilities,Process).

compatible(capabilities{dt:DT1},process{dt:DT2}):- DT2 < DT1,!.
compatible(capabilities{sf:SF1},process{sf:SF2}):- SF2 < SF1,!.
compatible(capabilities{tp:TP1},process{tp:TP2}):- TP2 < TP1,!.
compatible(capabilities{cy:CY1},process{cy:CY2}):- CY2 < CY1.

operation_compatible(Node1,Node2):-
   Node1 = 1 -> Node2 \= 4; true.

print_result([H|T]):-
   writeln(H),
   print_result(T).
print_result([]).
```

Modèles et méthodes pour la génération de processus de fabrication reconfigurables

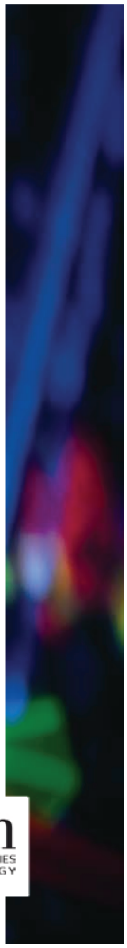
RESUME : Les approches conventionnelles pour la génération de processus de fabrication sont inefficaces pour gérer la complexité induite par la variété des produits et la dynamique de fabrication. La génération des processus de fabrication reconfigurables (RPP) est une nouvelle méthode de CAPP qui vise à générer des processus de fabrication pour une famille de produits / pièces. Cette thèse vise à apporter une contribution majeure aux modèles de représentation et aux méthodes de génération permettant la génération des processus de fabrication reconfigurables à deux niveaux de granularité : famille de produits et famille de pièces. Les approches proposées pour le RPP sont compatibles avec une extension du concept de famille de produits/pièces qui est défini en utilisant le concept de domaine. Un modèle pour une variété fonctionnelle de produit/pièce basé sur des entités est développé afin de représenter les informations nécessaires à la gestion des processus reconfigurables en utilisant des techniques modulaires. Des modèles mathématiques et des modèles de représentation sont proposés pour décrire le processus de fabrication reconfigurable à deux niveaux de granularité. Sur la base des modèles de représentation, les méthodes de génération et les algorithmes sont ensuite détaillés dans ce cadre. En outre, un cadre global est proposé pour décrire comment les modèles et les méthodes proposés collaborent pour gérer la variété de produit/pièce et la dynamique de fabrication. Pour illustrer la faisabilité des modèles et méthodes proposés, deux familles de pompes (l'une à engrenage et l'autre à huile) sont utilisées à titre d'exemples illustratifs tout au long de ce mémoire de thèse.

Mots clés : Personnalisation de masse, Génération des processus de fabrication, Famille de produits, Famille de pièces, Modèles de représentation, Modèles mathématiques

Representation models and generation methods to support reconfigurable process planning for product/part variety

ABSTRACT : Conventional manufacturing process planning approaches are very inefficient to handle the process planning complexity induced by product variety and manufacturing dynamics. Reconfigurable process planning (RPP) is an emerging CAPP approach targeting to the generation of process plans for a product/part family. This thesis aims to give major contributions to the representation models and generation methods to support reconfigurable process planning at two granularity levels: product family and part family. The proposed approaches for RPP are compatible with an extended concept of product/part family which is defined by using the concept of "domain". A feature-based product/part variety model is developed in order to represent the necessary information for RPP by using modular and platform-based techniques. Mathematical models and graph-based representation models are proposed to describe the reconfigurable process plan at two granularity levels. Based on the representation models, the generation methods and algorithms are then developed for RPP. In addition, a global framework is proposed to describe how the proposed RPP models and methods work together to handle the product/part variety and manufacturing dynamics. To test the feasibility of the proposed models and methods, a gear pump family and an oil pump body family are used as illustrative examples throughout this thesis.

Keywords : Mass Customization, Computer-aided Process Planning, Product family, Part family, Mathematical models, Representation models



ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY

Modèles et méthodes pour la génération de processus de fabrication reconfigurables

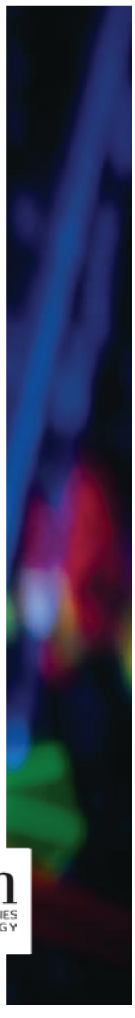
RESUME : Les approches conventionnelles pour la génération de processus de fabrication sont inefficaces pour gérer la complexité induite par la variété des produits et la dynamique de fabrication. La génération des processus de fabrication reconfigurables (RPP) est une nouvelle méthode de CAPP qui vise à générer des processus de fabrication pour une famille de produits / pièces. Cette thèse vise à apporter une contribution majeure aux modèles de représentation et aux méthodes de génération permettant la génération des processus de fabrication reconfigurables à deux niveaux de granularité : famille de produits et famille de pièces. Les approches proposées pour le RPP sont compatibles avec une extension du concept de famille de produits/pièces qui est défini en utilisant le concept de domaine. Un modèle pour une variété fonctionnelle de produit/pièce basé sur des entités est développé afin de représenter les informations nécessaires à la gestion des processus reconfigurables en utilisant des techniques modulaires. Des modèles mathématiques et des modèles de représentation sont proposés pour décrire le processus de fabrication reconfigurable à deux niveaux de granularité. Sur la base des modèles de représentation, les méthodes de génération et les algorithmes sont ensuite détaillés dans ce cadre. En outre, un cadre global est proposé pour décrire comment les modèles et les méthodes proposés collaborent pour gérer la variété de produit/pièce et la dynamique de fabrication. Pour illustrer la faisabilité des modèles et méthodes proposés, deux familles de pompes (l'une à engrenage et l'autre à huile) sont utilisées à titre d'exemples illustratifs tout au long de ce mémoire de thèse.

Mots clés : Personnalisation de masse, Génération des processus de fabrication, Famille de produits, Famille de pièces, Modèles de représentation, Modèles mathématiques

Representation models and generation methods to support reconfigurable process planning for product/part variety

ABSTRACT : Conventional manufacturing process planning approaches are very inefficient to handle the process planning complexity induced by product variety and manufacturing dynamics. Reconfigurable process planning (RPP) is an emerging CAPP approach targeting to the generation of process plans for a product/part family. This thesis aims to give major contributions to the representation models and generation methods to support reconfigurable process planning at two granularity levels: product family and part family. The proposed approaches for RPP are compatible with an extended concept of product/part family which is defined by using the concept of "domain". A feature-based product/part variety model is developed in order to represent the necessary information for RPP by using modular and platform-based techniques. Mathematical models and graph-based representation models are proposed to describe the reconfigurable process plan at two granularity levels. Based on the representation models, the generation methods and algorithms are then developed for RPP. In addition, a global framework is proposed to describe how the proposed RPP models and methods work together to handle the product/part variety and manufacturing dynamics. To test the feasibility of the proposed models and methods, a gear pump family and an oil pump body family are used as illustrative examples throughout this thesis.

Keywords : Mass Customization, Computer-aided Process Planning, Product family, Part family, Mathematical models, Representation models



ARTS ET MÉTIERS ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY