



Title Feature technology and its applications in
 computer integrated manufacturing

Name Lian Ding

This is a digitised version of a dissertation submitted to the University of Bedfordshire.

It is available to view only.

This item is subject to copyright.

FEATURE TECHNOLOGY AND ITS
APPLICATIONS IN COMPUTER INTEGRATED
MANUFACTURING

by
LIAN DING

A Thesis submitted for the degree of Doctor of Philosophy
of University of Luton

September 2003

kept at Enquiry Desk

UNIVERSITY OF LUTON PARK SQ. LIBRARY	
3402926928	
670.285	
DIN	

REFERENCE ONLY

Abstract

Computer aided design and manufacturing (CAD/CAM) has been a focal research area for the manufacturing industry. Genuine CAD/CAM integration is necessary to make products of higher quality with lower cost and shorter lead times. Although CAD and CAM have been extensively used in industry, effective CAD/CAM integration has not been implemented. The major obstacles of CAD/CAM integration are the representation of design and process knowledge and the adaptive ability of computer aided process planning (CAPP).

This research is aimed to develop a feature-based CAD/CAM integration methodology. Artificial intelligent techniques such as neural networks, heuristic algorithms, genetic algorithms and fuzzy logics are used to tackle problems. The activities considered include:

- 1) Component design based on a number of standard feature classes with validity check. A feature classification for machining application is defined adopting ISO 10303-STEP AP224 from a multi-viewpoint of design and manufacture.
- 2) Search of interacting features and identification of features relationships. A heuristic algorithm has been proposed in order to resolve interacting features. The algorithm analyses the interacting entity between each feature pair, making the process simpler and more efficient.
- 3) Recognition of new features formed by interacting features. A novel neural network-based technique for feature recognition has been designed, which solves the problems of ambiguity and overlaps.
- 4) Production of a feature based model for the component.
- 5) Generation of a suitable process plan covering selection of machining operations, grouping of machining operations and process sequencing. A hybrid feature-based CAPP has been developed using neural network, genetic algorithm and fuzzy evaluating techniques.

Acknowledgement

Firstly and most important, I would like to express my sincere gratitude to my supervisor Dr. Yong Yue for his valuable guidance, excellent supervision, support and continuous encouragement during this work. I have learnt many things from him.

I would like to express my gratitude to Dr. Kemal Ahmet for his continuous support, valuable comments and encouragement. I wish also to thank Dr. Jonathan R. Corney for his support and advices.

I wish to acknowledge the University of Luton and British Council for the financial support to carry out this project.

I would like to thank all members of the Faculty and the University who have assisted with this work.

I would like to thank my friend who has given me moral support with this work.

Last but not least, I want to extend special thanks to my parents and my sister who have given me continuing support and love during my doctoral studying.

Table of Contents

Abstract.....	II
Acknowledgement.....	III
Table of Contents.....	IV
List of Figures.....	XI
List of Tables.....	XIX
Notations.....	XXI
Declaration.....	XXIII
Chapter 1 Introduction.....	1
1.1. Background.....	1
1.1.1. Definition of features.....	2
1.1.2. Feature technology.....	3
1.1.3. Neural networks.....	4
1.1.4. CAPP.....	5
1.1.5. Genetic algorithms.....	6
1.2. Overview of this research.....	7

1.3	Organisation of the thesis	8
1.4	Thesis related publications	9
1.5	Summary.....	11
Chapter 2.	Literature Review	12
2.1	Conventional approaches of feature recognition.....	12
2.1.1	Graph matching method.....	13
2.1.2	Rule-base method	13
2.1.3	Volume decomposition method	13
2.1.4	Hybrid approach	14
2.2	Neural network-based feature recognition	14
2.2.1	The topology of artificial neural network	15
2.2.2	Input representation	18
2.2.3	The output format	24
2.2.4	The training method.....	25
2.3	Design by features	29
2.3.1	Design by features systems.....	29
2.3.2	Design by features techniques	32
2.4	Neural network-based CAPP.....	34

2.4.1	The topology of artificial neural network	35
2.4.2	Input representation	40
2.4.3	Output representation.....	43
2.4.4	Training method.....	44
2.5	CAPP using genetic algorithms.....	51
2.6	Summary.....	55
Chapter 3.	Design by Features Module.....	57
3.1	Architecture of the design by features module.....	57
3.2	Feature classification	60
3.3	Feature class definition.....	65
3.3.1	The requirements of feature class definition.....	65
3.3.2	The parameters of feature class definition.....	66
3.4	Feature-based model.....	73
3.5	Feature-based model management	75
3.5.1	Adding a new feature instance to the model.....	75
3.5.2	Editing a feature instance in the model.....	77
3.5.3	Deleting a feature instance from the model.	79
3.5.4	Checking validity of feature	79

3.6	Summary.....	87
Chapter 4. Interacting Features Recogniser.....89		
4.1	Basic terms and concepts.....	92
4.2	The types of feature interactions	96
4.3	Herustics algorithm for interacting feature recognition	97
4.4	Summary.....	108
Chapter 5. Ann-based Feature Recogniser.....109		
5.1	Design of neural network	109
5.2	Input representation	111
5.2.1	Attributed Adjacency Graph (AAG).....	111
5.2.2	Proposed input representation.....	113
5.3	Output format	125
5.4	The topology of neural network	127
5.5	Error function	132
5.6	Training of ANN	133
5.7	Summary.....	139

Chapter 6. Compute Aided Process Planning (CAPP).....	141
6.1. Requirements of CAPP.....	141
6.2. Architecture of CAPP.....	142
6.3. Resource management.....	145
6.4. Selection of machining operations.....	149
6.5. Identification of feature precedence list.....	152
6.6. Grouping of machining operations.....	156
6.6.1 Determination of TADs.....	157
6.6.2 Grouping algorithm.....	160
6.7. Summary.....	162
Chapter 7. Process Sequencing.....	163
7.1 Genetic Algorithm.....	165
7.1.1 Encoding scheme.....	165
7.1.2 Initial populations.....	169
7.1.3 Fitness function.....	173
7.1.4 Operators.....	176
7.1.5 Stop criteria.....	181
7.2 Evaluation of manufacturing rules for process sequence.....	182

7.2.1	Manufacturing sequence rules	183
7.2.2	Analytical hierarchy process (AHP).....	186
7.2.3	Construction of evaluating matrix	193
7.2.4	Evaluation result	194
7.3	Evaluation of time and cost	195
7.4	Weight calculation.....	200
7.4.1	Input representation	200
7.4.2	Output format.....	204
7.4.3	Topology and the training method of the neural network	204
7.5	Fuzzy evaluation of feature complexity	204
7.5.1	Fuzzy evaluation model.....	204
7.5.2	Feature evaluation.....	207
7.6	Summary.....	215
Chapter 8 Implementation and Testing of Prototype System		217
8.1	Facilities	217
8.2	System implementation	217
8.2.1	Phase I: Design by features.....	219
8.2.2	Phase II: Interacting features recognition	220

8.2.3	Phase III: ANN-based feature recogniser	221
8.2.4	Phase IV: CAPP	224
8.3	File management	226
8.4	Examples	227
8.4.1	Example 1	227
8.4.2	Example 2	236
8.4.3	Example 3	244
8.5	Summary.....	250
Chapter 9.	Conclusions.....	251
9.1	Research contributions	251
9.2	Limitations.....	254
9.3	Recommendations for future work	255
Reference	257
Appendix A	Feature Classification	270
Appendix B	Experiments of neural networks	280
Appendix C	Set theory	293
Appendix D	Flowcharts.....	297

List of Figures

Figure 1.1	Proposed architecture of CAD/CAM integration	8
Figure 2.1	Three-layer feed-forward neural network model.....	17
Figure 2.2	Four-layer feed-forward neural network model.....	18
Figure 2.3	Example of a Hopfield network.....	37
Figure 2.4	Example of MAXNET.....	40
Figure 3.1	Architecture of the design-by-features module	58
Figure 3.2	Example 1 of STEP classification	62
Figure 3.3	Example 2 of STEP classification	62
Figure 3.4	Definition of feature class.....	67
Figure 3.5	Geometric parameters of a hole	68
Figure 3.6	Geometric parameters of a slot	68
Figure 3.7	Example of interacting constraints	72
Figure 3.8	Example of feature Undigraph.....	72
Figure 3.9	Structure of feature model	74
Figure 3.10	Examples of feature relationships.....	76
Figure 4.1	Type I feature interaction.....	90
Figure 4.2	Type II feature interaction	91

Figure 4.3	Example of interacting features, $L_1 > L_2, d_1 < d_2$	91
Figure 4.4	Example of interacting features, $L_1 > L_2, d_1 = d_2$	92
Figure 4.5	Valid components for this research	93
Figure 4.6	Invalid component due to a sloping hole	93
Figure 4.7	Example of SVE, FF, VF and NF	94
Figure 4.8	Example of IE	95
Figure 4.9	Example of $IE_{AB} = \emptyset$	97
Figure 4.10	Example of ERI	98
Figure 4.11	Example of FRI (PF-PF)	98
Figure 4.12	Example of $IE_C = \text{face} \in \text{CF-PF}, L_1 = L_2$	99
Figure 4.13	Examples of $IE_C = \text{face} \in \text{CF-CF}, IE_C = \text{CF}_A, IE_C = \text{CF}_B$ (a) $L_1 = L_2, d_1 = d_2, w_1 = w_2$ (b) $L_3 = L_4, d_3 = d_4, w_3 = w_4$	100
Figure 4.14	Examples of $IE_C = \text{face} \in \text{CV-PF}, IE_C = \text{CV}_1, IE_C = \text{PF}_{II}$	100
Figure 4.15	Example1 of $IE_{AB} = \text{Volume}, f_i \in IE_{AB}, f_j \in IE_{AB}, f_i \in \text{CF-NF}, f_j \in \text{PF-PF}, L_1 = L_2, d_1 = w_2, h_1 = h_2$	102
Figure 4.16	Example2 of $IE_{AB} = \text{Volume}, f_i \in IE_{AB}, f_j \in IE_{AB}, f_i \in \text{NF-CF}, f_j \in \text{PF-PF}, L_1 + w_1 < L_2 + w_2, w_2 > w_1, h_1 = h_2$	102
Figure 4.17	Example of $IE_{AB} = \text{Volume}, f_i \in IE_{AB}, f_i \in \text{PF-CF}, L_1 = L_2$	103
Figure 4.18	Example of $IE_{AB} = \text{Volume}, f_i \in IE_{AB}, f_i \in \text{CF-NF}, L_1 < L_2, w_1 > w_2, h_1 > h_2$	104

Figure 4.19 Example of $IE_{AB} = \text{Volume}$, $f_i \in IE_{AB}$, $f_i \in CV\text{-}PV$ $d_1 > d_2$, $d_1 + L_1 < d_2 + L_2$, $h_1 < h_2$105

Figure 4.20 Example of $IE_{AB} = \text{Volume}$ with two non-connected *NFs* 105

Figure 5.1 Features with the same face-edge graph 112

Figure 5.2 A feature with two matrices..... 112

Figure 5.3 Examples of SVE, where P indicates a planar face 114

Figure 5.4 Examples of Depth-first search method..... 115

Figure 5.5 Pre-process steps..... 119

Figure 5.6 Pre-processing of Radiused Blind Slot 120

Figure 5.7 Simplification of topology 121

Figure 5.8 Values of relationship between two faces..... 123

Figure 5.9 Hierarchical neural networks system 127

Figure 5.10 The training process of neural networks (a) structure of 15-17-5

$$\beta_k = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \dots\dots\dots 129$$

Figure 5.11 Three-layer Feedforward neural network for level 1 recogniser 132

Figure 5.12 Illustration of the line search algorithm [Charalambous, 1992] 136

Figure 5.13 The training process of neural networks (continue) structure of

$$15\text{-}17\text{-}5 \quad \beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \dots\dots\dots 139$$

Figure 6.1 The proposed architecture of CAPP 143

Figure 6.2	The relationships between libraries	148
Figure 6.3	An example of feature-precedence graph	154
Figure 6.4	Procedure of grouping machining operations.....	161
Figure 7.1	The procedure for process sequencing.....	164
Figure 7.2	Data structure of evaluation criteria.....	184
Figure 7.3	The structure of weights collated.....	208
Figure 7.4	Examples of nominal dimensions, dimensional accuracy, surface roughness and tolerances	211
Figure 8.1	The block diagram of the prototype system.....	218
Figure 8.2	The design by features interface	220
Figure 8.3	The interface for input of feature parameters for a Through Slot....	220
Figure 8.4	Modelling of test component 1	229
Figure 8.5	Tolerance and surface finish of the features of test component 1 ...	229
Figure 8.6	Interacting entity between Feature 2 and Feature 3	230
Figure 8.7	Interacting entity between Feature 4 and Feature 5	231
Figure 8.8	Result of feature recognition for test component 1.....	233
Figure 8.9	Feature precedence list of test component 1	234
Figure 8.10	Machining operation groups test component 1	234
Figure 8.11	Process plan generated for test component 1	235
Figure 8.12	Modelling of test component 2	237

Figure 8.13 Interacting entity between Feature 1 and Feature 2	238
Figure 8.14 Interacting entity between Feature 4 and Feature 5	239
Figure 8.15 Tolerance and surface finish of the features of test component 2 ...	240
Figure 8.16 Feature recognition result of test component 2.....	241
Figure 8.17 Feature precedence list of test component 2	242
Figure 8.18 Machining operation groups test component 2.....	242
Figure 8.19 Process planning for test component 2	243
Figure 8.20 Modelling of test component 3	244
Figure 8.21 Tolerance and surface finish of the features of test component 3 ...	245
Figure 8.22 Feature recognition result of test component 3.....	247
Figure 8.23 Feature precedence list of test component 3.....	248
Figure 8.24 Machining operation groups test component 3.....	248
Figure 8.25 Process planning of test component 3.....	249
Figure B.1 The Polak-Ribiere conjugate gradient backpropagation training process of 15-17-5 neural network (Level 1).....	280
Figure B.2 The Fletcher-Powell conjugate gradient backpropagation training process of 15-17-5 neural network (Level 1).....	280
Figure B.3 Gradient descent backpropagation training process of 15-17-5 neural network (Level 1).....	281
Figure B.4 The Polak-Ribiere conjugate gradient backpropagation training process of 15-16-5 neural network (Level 1).....	281

Figure B.5 The Fletcher-Powell conjugate gradient backpropagation training process of 15-16-5 neural network (Level 1).....	282
Figure B.6 Gradient descent backpropagation training process of 15-16-5 neural network (Level 1).....	282
Figure B.7 The Polak-Ribiere conjugate gradient backpropagation training process of 15-18-5 neural network (Level 1).....	283
Figure B.8 The Fletcher-Powell conjugate gradient backpropagation training process of 15-18-5 neural network (Level 1).....	283
Figure B.9 Gradient descent backpropagation training process of 15-18-5 neural network (Level 1).....	284
Figure B.10 The Polak-Ribiere conjugate gradient backpropagation training process of 15-15-5 neural network (Level 1).....	284
Figure B.11 The Fletcher-Powell conjugate gradient backpropagation training process of 15-15-5 neural network (Level 1).....	285
Figure B.12 Gradient descent backpropagation training process of 15-15-5 neural network (Level 1).....	285
Figure B.13 The Polak-Ribiere conjugate gradient backpropagation training process of 15-14-5 neural network (Level 1).....	286
Figure B.14 The Fletcher-Powell conjugate gradient backpropagation training process of 15-14-5 neural network (Level 1).....	286
Figure B.15 Gradient descent backpropagation training process of 15-17-5 neural network (Level 1).....	287
Figure B.16 The Polak-Ribiere conjugate gradient backpropagation training process of 21-1-2 neural network (Round Hole)	287

Figure B.17 The Polak-Ribiere conjugate gradient backpropagation training process of 21-2-2 neural network (Round Hole)	288
Figure B.18 The Polak-Ribiere conjugate gradient backpropagation training process of 21-3-2 neural network (Round Hole)	288
Figure B.19 The Fletcher-Powell conjugate gradient backpropagation training process of 21-2-2 neural network (Round Hole)	289
Figure B.20 The Polak-Ribiere conjugate gradient backpropagation training process of 21-6-4 neural network (Slot/Step)	289
Figure B.21 The Polak-Ribiere conjugate gradient backpropagation training process of 21-5-4 neural network (Slot/Step)	290
Figure B.22 The Fletcher-Powell conjugate gradient backpropagation training process of 21-6-2 neural network (Slot/Step)	290
Figure B.23 The Polak-Ribiere conjugate gradient backpropagation training process of 21-2-2 neural network (Pocket)	291
Figure B.24 The Polak-Ribiere conjugate gradient backpropagation training process of 21-1-2 neural network (Pocket)	291
Figure B.25 The Fletcher-Powell conjugate gradient backpropagation training process of 21-1-2 neural network (Pocket)	292
Figure D.1 Process of adding a new feature instance	297
Figure D.2 Process of editing feature	298
Figure D.3 Process of deleting feature	299
Figure D.4 Interacting feature identify algorithm	300
Figure D.5 Situation II, IE_C = face, interacting feature recogniser	301

Figure D.6	Situation III, $IE_C=Volume$, interacting feature recogniser	302
Figure D.7	Process of selecting machining operations	303
Figure D.8	Process of building a feature precedence list.....	304
Figure D.9	Procedure for generating a tool approach direction.....	305
Figure D.10	Diagram of the proposed genetic algorithm process	306
Figure D.11	Initial precedence constraint algorithm.....	307
Figure D.12	The proposed AHP model for evaluation	308

List of Tables

Table 2.1.	Capabilities of ANN-based feature recognition systems.....	28
Table 2.2.	Achievements of ANN-based CAPP systems	50
Table 3.1	Classification of machining features.....	64
Table 3.2	Calculating expressions for blind slot.....	80
Table 3.3	Calculating expressions for through slot	82
Table 3.4	Calculating expressions for through step.....	84
Table 3.5	Calculating expressions for closed pocket.....	85
Table 3.6	Calculating expressions for blind/through hole	87
Table 4.1.	Face types for IE	95
Table 4.2.	Relationship between interacting features	106
Table 5.1	Value of face type	116
Table 5.2	Examples of F-adjacency Matrix	124
Table 5.3	Examples of V-adjacency Matrix.....	126
Table 6.1	An example of building feature precedence list.....	157
Table 7.1.	Example of Initial precedence constraint algorithm	174
Table 7.2.	Evaluating criteria for the pairwise comparison matrix.....	187
Table 7.3.	The values of RI	192

Table 7.4.	Evaluation for feature complexity	201
Table 7.5.	An example of input neurons 1 to 20.....	202
Table 7.6.	The evaluating values for production batch size	203
Table 7.7.	The evaluating values for production urgency	203
Table 7.8.	Grades for round holes.....	209
Table 7.9.	\bar{u}_{type} of round holes.....	209
Table 7.10.	\bar{u}_{type} of slot.....	209
Table 7.11.	\bar{u}_{type} of step	209
Table 7.12.	\bar{u}_{type} of pocket	210
Table 7.13.	Fuzzy value for surface roughness.....	212
Table 8.1	Examples of training features	222
Table 8.2	Design.mdb	224
Table 8.3	Manufacturing library.mdb	225

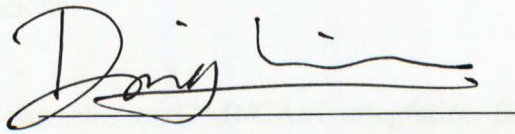
Notations

CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
S	Stock
f_{t_i}	Machining feature
SVE	Spatial Virtual Entity
FF :	Feature Face
PF	Partial FF
CF	Whole FF
VF	Virtual Face
PV	Partial VF
CV	Whole VF
NF	None Face
IE	Interacting Entity
\cup	Union
\cap	Intersection
NS	Node Sequence
N_f	Number of adjacent faces
N_v	Number of adjacent virtual faces
$T_{f_{type}}$	Face type value
OAL	Ordered Adjacency List,
$USet$	A set of faces that have not been visited yet
X_i	The i th training pattern.
$W^{(0)}$	Initial weight vector
d_0	Steepest descent direction
g_0	Negative of the gradient
$E(W)$	Error function
α_k	Learning rate
PR	A set of all feasible process routes based on feature class

<i>TAD</i>	Tool Approach Direction
<i>MT</i>	Machining time
<i>FPR</i>	A set of all feasible process routes based on design specifications
<i>L</i>	Feature precedence list
<i>UTAD</i>	A set of features whose TADs have not been determined yet
<i>FT_i</i>	A set of TAD candidates that the <i>i</i> th feature
λ_{\max}	maximum Eigenvalue of R-matrix
<i>CR</i>	inconsistency ratio
<i>CI</i>	consistency index of R-matrix
<i>RI</i>	random consistency index of R-matrix
<i>V</i>	V-matrix
<i>CM</i>	Machine cost
<i>CT</i>	Tool cost
<i>CMC</i>	Machine change cost
<i>CTC</i>	Tool change cost
<i>CSC</i>	Setup change cost
<i>C</i>	total cost for the process plan.
<i>GC</i>	Machining cost for operation group.
<i>GCC</i>	Changing cost for two adjacent groups
<i>C_{max}</i>	Maximum manufacturing cost for the component
<i>TM</i>	Machining time
<i>TMC</i>	Machine change time
<i>TTC</i>	Tool change time
<i>TSC</i>	Setup change time
<i>GT</i>	Machining time for operation group
<i>GTC</i>	Changing time for two adjacent groups
<i>T_{max}</i>	Longest manufacturing time for the component
<i>U</i>	Fuzzy evaluation matrix
\bar{U}	Fuzzy vector
<i>R_a</i>	Roughness average

Declaration

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Doctor of Philosophy at the University of Luton. It has not been submitted before for any degree or examination in any other University.

A handwritten signature in black ink, appearing to read 'Lian Ding', written over a horizontal line.

LIAN DING (CANDIDATE)

Chapter 1

Introduction

1.1. Background

With the growing trend towards global market, industry is facing fierce competition. Traditional design and manufacturing practice is no longer suitable for the new requirements. It has been widely recognised that genuine integration of design and manufacturing is needed to make products of higher quality with lower cost and shorter lead times. Although CAD and CAM have been extensively used in industry, effective CAD/CAM integration has not been implemented and human intervention is often required to interpret design data for downstream applications.

One of the major obstacles of CAD/CAM integration is the representation of design and process knowledge. Geometrical models only provide the geometric and topological information of a component, which is not sufficient for manufacturing applications, e.g. process planning. Thus, features encapsulating the engineering significance are considered as a key element in the integration of design and manufacturing and feature-based models have been widely used. Although considerable work has been done in feature technology, progress has been hindered by interacting features and inability of self-learning for feature recognition.

Computer-aided process planning (CAPP) is considered as another key element for CAD/CAM integration, which automates to some extent, the decision making

in process selection, sequencing and parameter calculation. A considerable number of CAPP systems have been developed, but only a few of them have the abilities of adaptation and self-learning for certain tasks.

1.1.1. Definition of features

A feature cannot be defined without considering its applications. That is, features are application dependent. Numerous definitions of features according to the application are given by various researchers, such as “any geometric form or entity uniquely defined by its boundaries, or any uniquely defined geometric attribute of a part that is meaningful to any life cycle issue” [Dixon *et al*, 1987]; “regions of a part having some machining significance” [Joshi and Chang, 1988]; “solid removable by operations typically performed in a 3-axis machining centre” [Vandenbrande and Requicha, 1990], “a set of geometric entities (faces, edges and vertices) together with specifications of the bounding relationships between them that together imply an engineering function on an object” [Kang and Nnaji, 1993]; “a feature represents the engineering meaning or significance of the geometry of a part or assembly” [Shah and Mäntylä, 1995], “form features can be defined as a part geometry associated with process planning entities such as slots and pockets” [Zhang *et al*, 1997], “a geometric feature is traditionally defined as any subset of the geometric model of the object that is of interest in a particular context (e.g. in CAPP)” [Yue and Venuvinod, 1999]. From the design and manufacturing point of view, features referred to in this thesis is defined as a

geometrical entity, which may be associated with a group of particular machining processes and can be used to reason about a suitable machining method.

1.1.2. Feature technology

Feature technology has been considered an indispensable tool for integrating design and manufacturing processes. Feature recognition and design by features are the two major approaches to creating feature models [Bronsvort and Jansen 1993]. Feature recognition makes direct use of geometric models and generates application-specific feature models using various recognition rule sets regarding the application. A principal advantage of the feature recognition is the possibility of using conventional CAD systems directly. However, there are problems with feature recognition such as feature interactions hindering its practical applications. With a design by features approach, the designer specifies a design model using a set of design features defined in a feature-based model system [Lee and Kim 1998]. In contrast to feature recognition, design by features can capture the design and manufacturing information during the design stage. It reduces remarkably the amount of work for recognising features, but does not eliminate the need for feature recognition [Gindy *et al* 1998]. Thus feature recognition techniques are required in all systems that use features for analysis and decision making.

1.1.3. Neural networks

Artificial neural networks (ANNs) are information processing devices consisting of many interconnected processing elements (neurons) based on the neural structure of the brain. An ANN is configured for a specific application by a learning process, such as classification, pattern recognition, optimisation or prediction. Neural network techniques regarded as an adaptive method has advantages on the applications of both feature recognition and CAPP:

- 1) A neural network can tolerate slight errors from input;
- 2) The techniques are faster because the process is limited to simple mathematical computations and does not use either a search or logical rules to parse information;
- 3) An ANN feature recogniser possesses experience to recognise and classify similar features since it is trained and there is no need to predefine every instance of a feature as in most traditional systems.

Although much has been done with various approaches and certain success achieved in feature recognition and CAPP, most of the methods do not have the learning capabilities of neural networks. Neural network techniques which have been prevalent in recent years, offer a new promising solution in these areas of research.

1.1.4. CAPP

Process planning establishes a set of manufacturing operations and their sequence, and specifies the appropriate tools and process parameters in order to produce a component from its initial raw state to a final form predetermined from an engineering drawing. The use of computer techniques to automate the tasks of process planning - computer aided process planning (CAPP) has been the subject of extensive research for CAD/CAM integration. A CAPP system usually performs the determination of machining operations, selection of suitable setups and machining resources, and process sequencing. As a key technology for computer aided design and manufacturing integration, CAPP strongly influences the cost of production and the quality of a product. The greater the degree of automation of a CAPP system, the shorter the time from design to machining, and the better the quality of the final product owing to the elimination of human error [Yip-Hoi and Dutta, 1996].

Two approaches have been developed in CAPP: variant and generative. Variant process planning systems use group technology concepts to aid a process planner. Features sharing common manufacturing characteristics are classified into the same group. This approach may cut down process planning time dramatically, especially for similar components. However, there is a main drawback that process plans are limited to those that were previously created. In generative process planning systems, a process plan is created by utilising rules of expert knowledge. In comparison to the variant method, the generative method needs less human intervention and new parts may be planned as easily as existing

components. The main problem is that it cannot adapt to change in manufacturing practice and technology. Further, expert knowledge acquisition is time-consuming, costly and error-prone.

1.1.5. Genetic algorithms

A genetic algorithm (GA) being one of the most popular combinatorial algorithms and artificial intelligence (AI) technique, is a search technique for solving optimisation problems based on the mechanics of the survival of the fittest [Dereli and Filiz, 1999]. The algorithm incrementally converges to an optimal or near-optimal solution based on a series of biological operations including selection, crossover and mutation. A GA has certain favourable characteristics, which make it an attractive tool for use in process sequencing:

- 1) It is relatively easy to adapt for process planning due to its characteristics such as suitable encoding scheme and effective fitness function.
- 2) It is flexible in improving on poor performance by varying its input parameters including the initial population size and mutation rate.
- 3) It permits a straightforward amendment of the heuristic embodying in the fitness function of the algorithm.

1.2. Overview of this research

The aim for this research is to develop a feature-based CAD/CAM integration methodology. Specific objectives include a) a novel feature-based modelling approach, b) machining feature extraction by solving interacting features individually, c) intelligent CAPP based on the features extracted. The work consists of four main stages shown in Figure 1.1.

- 1) The first stage uses a design by features approach to build a feature-based model of a component. Based on standard feature library, it maintains the feature-based model automatically.
- 2) The second stage applies feature recognition techniques to extracting machining features from the feature-based model. The focus is on interacting features. Incorporating a heuristic algorithm, a neural network-based feature recogniser is used to recognise and resolve interacting features.
- 3) The third stage carries out the task of process planning based on the machining features extracted. It starts with the application of the feature-based model and uses the extracted machining feature information directly for selecting and grouping machining operations, and optimising process sequence.
- 4) The fourth stage builds and enhances the system's capability through incremental learning.

Feature recognition, design by features, neural network techniques, genetic algorithms, and fuzzy evaluation are employed aiming at resolving feature interactions and automating process planning.

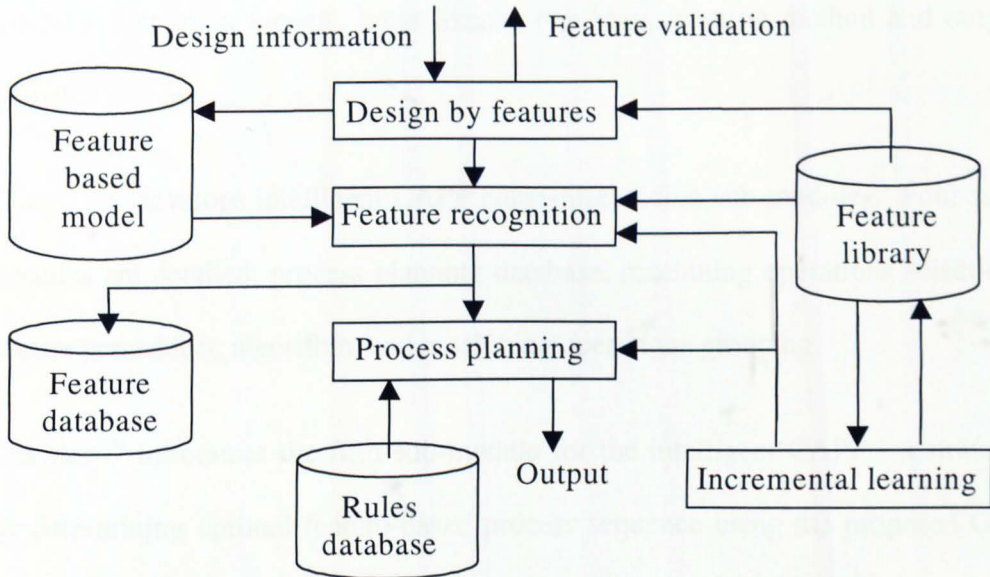


Figure 1.1 Proposed architecture of CAD/CAM integration

1.3 Organisation of the thesis

The thesis is presented in nine chapters as follows.

Chapter 1 introduces the background and scope of the research.

Chapter 2 reviews the work in conventional and ANN-based feature recognition, design by features, ANN-based CAPP and CAPP using genetic algorithms.

Chapter 3 proposes a design by features approach and its system organisation. Feature classification, feature library, feature-based model and feature-based model management are described.

Chapter 4 presents a novel heuristic algorithm for the identification, analysis and processing of interacting features.

Chapter 5 describes the neural network techniques adopted in the research, covering four main aspects: input format, topology, learning method and output format.

Chapter 6 develops intelligent CAPP consisting of five sub-modules. Four sub-modules are detailed: process planning database, machining operations selection, feature precedence algorithm, and machining operations grouping.

Chapter 7 introduces the fifth sub-module for the intelligent CAPP - a strategy for determining optimal feature-based process sequence using the proposed GA, analytical hierarchical process, estimation of cost and time of a process plan and allocation of relative weights by a neural network and fuzzy evaluation.

Chapter 8 implements proposed methodology, and presents the testing results with a range of components.

Chapter 9 discusses the contributions and limitations, and suggests recommendations for future work for the research.

1.4 Thesis related publications

This thesis presents the author's original work except for the acknowledged where appropriate. Some of the work described here has been published previously in journal or conferences. These include:

- Ding, L., Yue, Y. and Ahmet, K., “A Novel Input Representation for ANN-Based Features Recognition”, *Frontiers in Artificial Intelligence and Applications*, Vol 82, *Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies*, KES 2002, Part I, pp 311-315, 2002
- Yue, Y. Ding, L., Ahmet, K., Painter, J. and Walters, M., “Study into Neural Network Techniques for Computer Integrated Manufacturing”, *Engineering Computations*, Vol. 19, No. 1-2, pp. 136-157, 2002
- Ding, L. and Yue, Y., “An ANN approach to Feature Recognition”, *Proceedings of the 8th Conference of the Chinese Automation and Computer Society in the UK and CASIA Annual conference on Automation and Information Technology*, pp.26-31, Beijing, 21-22 September 2002
- Ding, L. and Yue, Y., “An Intelligent Hybrid Approach for Design-by-Features”, *Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Plzen, Czech Republic, pp. 4-8, 2-4 February 2002
- Ding, L., Yue, Y. and Ahmet, K., “Artificial Neural Network Applications to Process Planning”, *Proceedings of the 2001 International Conference on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, Vol. II, pp. 778-783, June 2001
- Ding, L., Yue, Y. and Ahmet, K., “An integrated approach to integrating CAD/CAM”, *6th Conference of the Chinese Automation and Computer Society in the UK*, pp. 91-96, Loughborough, 23-24 September 2000

- Ding, L., Yue, Y., Painter, J. and Walters, M., “Artificial neural network applications to feature recognition”, *16th National Conference on Manufacturing Research*, pp. 41-46, London, 5-7 September 2000
- Yue, Y., Ding, L. and Ahmet, K., “Neural Network Systems Technology and Applications in Computer Integrated Manufacturing”, *Business and Technology of the New Millennium*, Cornelius T. Leondes (ed), Kluwer Academic Press, Accepted, 2003

A further journal paper is under preparation for submission shortly.

1.5 Summary

This chapter has given a brief introduction of the author’s research, including the background, scope, organisation of the thesis and the related publications. The research will be described in detail in the next eight chapters.

Chapter 2.

Literature Review

There are two key issues with CAD/CAM integration: representation of design information, and acquisition and representation of process knowledge, especially empirical knowledge, which is useful for analysis and decision making. Features encapsulate the engineering significance of portions of the product geometry and, as such, are applicable in product design, product definition, and reasoning about the product in a variety of applications such as manufacturing planning [Shah and Mäntylä, 1995]. Thus, features have been used as a means of interface in computer integrated manufacturing (CIM) through computer aided process planning (CAPP), and feature technology has been considered an indispensable tool for integrating design and manufacturing processes. Feature recognition and design by features are the two major approaches to creating feature models [Bronsvoort and Jansen 1993]. This chapter reviews the work related to the feature recognition, design by features and process planning systems.

2.1 Conventional approaches of feature recognition

Feature recognition generates an application-specific feature model using various recognition rule sets, by searching the component geometric model. It is a preferred method but there are problems such as feature interactions and high

algorithmic complexity. Various feature recognition methods have been proposed in the last decade. There have been four common approaches.

2.1.1 Graph matching method

Graph matching method organises a B-rep model of a part into a stereotypical sub-graphs structure where the nodes represent faces, edges or vertices and the arcs represent the relationships of any two entities. Joshi and Chang [1988], De Floriani and Bruzzone [1989], Lentz and Sowerby [1993] have pursued this method. The graph-based recognition approach has an advantage over the others due to the graph nature of B-rep-based solid model [Lam and Wong, 2000]. It is effective, but suffers from two significant drawbacks: the large computational expenditure of dealing with complex components, and the deficiency of dealing with interacting features.

2.1.2 Rule-base method

Rule-base method uses artificial intelligence techniques to develop a set of feature rules. Choi *et al* [1984], and Donaldson and Corney [1993] have employed this approach. The major difficulty with the rule-based method is that it is impossible to define all rules for manufacturing features in reality [Wong and Lam, 2000].

2.1.3 Volume decomposition method

Volume decomposition method divides the three-dimensional space surrounding an object into cells with all the geometric surfaces of an object. Several researchers, such as Woo [1982], Tang and Woo [1991], Kim [1992], and Woo and Sakurai [2002], have adopted this approach. The process of volume

decomposition method is complex and the computational cost of the method is high. In addition, two aspects still need further efforts, which are multiple feature interpretations for interacting features and the limited recognising domain due to the problems of non-converging decomposition.

2.1.4 Hybrid approach

Hybrid approach combines several basic techniques to increase the recognition power [Shah and Mäntylä 1995]. There have been instances of work using this approach, such as Vandenbrande and Requicha [1990], Laakko and Mantyla [1991], Sreevalsan and Shah [1992], Lam and Wong [2000] and Miao *et al* [2002]. Hybrid approaches enhance the ability to handle feature interactions, but considerable efforts are still needed.

Although the above methods have their own advantages and limitations, there are four principal drawbacks with conventional feature recognition approaches that can be summarised as: the inability to recognise inexact or incomplete features, slow execution speed, inability to recognise all interacting features completely and correctly, and the inability to learn.

2.2 Neural network-based feature recognition

An ANN is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron [Gurney 1997]. The function of an ANN-based system is determined by four parameters: the net topology, training or learning rules, input node characteristics and output node

characteristics [Prabhakar and Henderson 1992].

Neural network-based methods can eliminate some drawbacks of the conventional approaches, and therefore have attracted research attention particularly in recent years. This section discusses the following aspects of ANN applications to feature recognition: the network topology, input representation, output format, training or learning method, and a summary of the results.

2.2.1 The topology of artificial neural network

There are three main ANN architectures: feed-forward, recurrent and competitive networks [Gurney 1997]. An ANN applied to feature recognition is generally a feed-forward model which is a layered network, either fully interconnected from layer to layer or containing hidden units. The typical topology can be defined by an input layer of neurons that receive binary or continuously valued input signals, an output layer with a corresponding number of neurons, and a number of hidden layers that are highly interconnected [Nezis and Vosniakos 1997]. Hence, the design of network topology can be reduced to three problems: the number of hidden layers, the optional number of neurons in the hidden layer, and the use of networks with incompletely connected layers. At present, three main feed-forward architectures have been developed for feature recognition as described below.

1) The five-layer, perceptrons quasi-neural network

Prabhakar and Henderson [1992] developed a five-layer, perceptrons quasi-neural network system called PRENET. The system has five layers which

respectively, consist of N nodes, N groups of M nodes, N nodes with a threshold non-linearity, M nodes corresponding to the M conditions for a feature, and one node, where N is the number of faces in the test part and M is the number of conditions required for the feature. Corresponding to the network architecture, there are four steps in the recognition process:

- converting the input vectors of a row into single integers called codes,
- searching for the integers corresponding to the feature definition,
- finding the faces satisfying the conditions specified for a feature, and
- producing the recognition result to the node in the 5th layer by an AND operation.

2) The three-layer feed-forward neural network

This model has an input, a hidden and an output layer (shown in Figure 2.1). Neurons on the hidden and output layers are defined from the neurons on the previous layer, the weights and a processing algorithm. For example, in Chuang's system [Chuang *et al* 1999], the l th neuron on the current layer, N_l can be calculated as:

$$N_l = \sum_{k=1}^n u_k w_{kl}, \quad (2-1)$$

where u_k is the k th neuron on the previous layer, and

w_{kl} is the weight representing the strength of the relationship

between the k th neuron on the previous layer and the l th neuron

on the current layer.

In order to constrain the value of each neuron on the current layer ranging from 0 to 1, a sigmoid function is used as a transfer function:

$$F(N_i) = \frac{1}{1 + e^{-N_i}} \quad (2-2)$$

Further, for the neuron on the output layer, the value is converted into 0 or 1 by an appropriate thresholding scheme. There have been other instances of using three-layer feed-forward neural networks, such as Peters' work [1992] on 2D feature recognition and Hwang's work [1991] on 3D feature recognition.

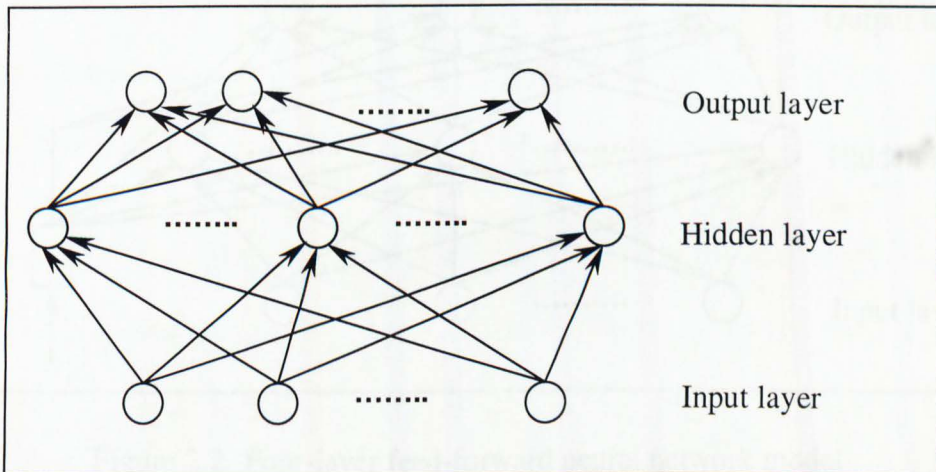


Figure 2.1 Three-layer feed-forward neural network model

3) The four-layer feed-forward neural network

This approach uses four layers: an input, a hidden, an output, and a threshold layer, which is added to the network as the training is completed. The threshold layer performs the function of activating the neurons of the output

layer by a threshold, e.g. 0.5. In Nezis and Vosniakos' work [1997], there are 20 neurons in the input layer, each representing an element of the input vector, and eight output neurons, each corresponding to a feature class. There are also eight neurons in the threshold layer, corresponding to the output layer neurons. There are also eight neurons in the hidden layer, corresponding to the output layer neurons. Ten neurons are assigned to the hidden layer by experimentation. All elements of the hidden and output layers are connected with a bias element that can be considered as an activation threshold. The topology of the neural network in Nezis' system is shown in Figure 2.2. The other example of this architecture is the work of Öztürk N. and Öztürk F. [2001].

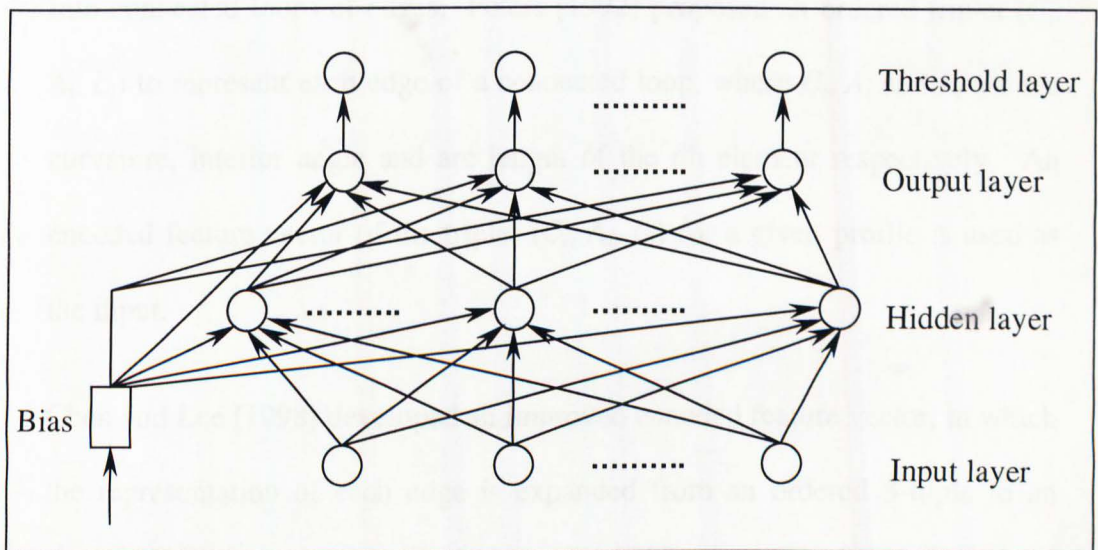


Figure 2.2 Four-layer feed-forward neural network model

2.2.2 Input representation

Neural nets typically, although not necessarily, receive a set of integer values. The problem then is how to convert a solid model to a format suitable for neural net input in a convenient and efficient way. There are three basic characteristics for a satisfactory input representation:

- 1) the solid model information (e.g. faces, edges and vertices) for feature recognition,
- 2) a format identifiable by the input layer, and
- 3) a unique input representation without overlaps.

The input representation can be broadly classified into the following types:

- 1) 2D feature representation

In engineering drawings, the wire-frame profiles of shapes can be subdivided into connected loops of edges. Peters [1992] proposed an ordered triplet (C_i, A_i, L_i) to represent each edge of a connected loop, where C_i , A_i and L_i are the curvature, interior angle and arc length of the i th element respectively. An encoded feature vector of the triplet (C_i, A_i, L_i) for a given profile is used as the input.

Chen and Lee [1998] developed an improved encoded feature vector, in which the representation of each edge is expanded from an ordered 3-tuple to an ordered 7-tuple in the form: $(L_i, A_i, C_i, J_i, OL_i, OA_i, OC_i)$ where J_i is the intersection type between the line segment and its subsequent line segment, and OL_i , OA_i and OC_i are the ordinal values assigned to L_i , A_i and C_i respectively. The ordinal values are assigned to the parameter in order to capture the magnitudes. The input layer has thirty-five neurons corresponding to five edges, seven neurons representing each edge. Although this method can recognise three and four-sided features, more neurons are needed in the input, output and hidden layers when the number of edges of a feature is

increased.

2) Face adjacency matrix code

A face adjacency matrix is a 2D array of integer vectors converted from a solid model. Each integer vector represents a face and its relationship to another face, i.e. adjacency or common edge. The length of an integer vector depends on the number of parameters considered for the recognition of a feature. In Prabhakar and Henderson's work [1992], the vector has eight integers indicating characteristics such as edge type, face type, face angle type, number of loops, etc. This method is limited to features defined by a primary face and a set of secondary faces. It cannot differentiate between features with the same topology but different dimensions of compound faces.

3) Face score vector

This represents the relationship between the main face of a feature and its neighbouring faces [Hwang 1991]. The eight-element face score vector is formed in three steps.

- A face score is defined as $F_s = f(F_g, E_g, V_g, A_t)$, where F_s is the face score, F_g , E_g and V_g are the information about the face, edge and vertex geometry, and A_t is the adjacency among the faces, edges and vertices. A high face score indicates a likely feature face, which in turn indicates the addition or removal of material.
- A face score graph representing the relationship of face scores between a face and its neighbouring faces is drawn based on the face scores for all

faces of the given object. A non-zero difference between a face score and its neighbouring face score indicates a geometric or topological change between these faces, which form a region. The region may be defined as a feature.

- An eight-element face score vector is formed and input to the net. Chan's work [1994] is another example of a face score vector while Srinath [1993] tackled partial features.

This representation can recognise a very limited number of compound features, and there is no one-to-one correspondence between feature patterns and features.

4) Attributed adjacency matrix

An attributed adjacency matrix [Nezis and Vosniakos 1997 and Gu *et al* 1995] describing the geometry and topology of a feature pattern is converted from the attributed adjacency graph (AAG) [Joshi and Chang 1988]. In Nezis and Vosniakos' research [1997], the adjacency matrix (AM) is a 2D, square, binary matrix with two triangular areas: an upper and a lower which are the convex and concave spaces respectively. $AM[i, j]$ and $AM[j, i]$ indicate the connection between the i th and j th faces of the object. One of them belongs to the concave space and the other to the convex space. The representation vector is formed as follows:

- the AAG is broken into sub-graphs which are converted into AM using a heuristic method;

- each matrix is converted into a representation vector (RV) by interrogating a set of 12 questions about the AM layout and the number of faces in the sub-graph; and
- a binary vector is formed combining the 12 positive answers and the other 8 elements corresponding to the number of external faces linked to the sub-graph.

This method can recognise planar and simple curved faces, but not features related to secondary feature faces, such as T-slots.

5) 2D input patterns of 3D feature volume

Zulkifli and Meeran [1999] presented an input matrix based on a cross-sectional method. The B-rep solid model is searched through cross-sectional layers and converted into 2D feature patterns, which are then translated into a matrix appropriate to the network. Four input matrices correspond to four feature classes: simple primitive, circular, slanting, and non-orthogonal primitive features. There are several disadvantages, e.g. simple primitive features are limited to four rectangular vertices, and features with non-orthogonal faces in the z direction cannot be dealt with.

6) A vector based on the partitioned view-contours of a given object

The given object is represented by nine partitioned view-contours from +x, -x, +y, -y, +z, -z, x, y and z respectively. The vector is built in three steps.

- A graph with a representative ring code is defined from a partitioned view-contour in which the nodes represent the regions and the arcs the adjacency relations among the regions; the representative ring code is a cyclic string of digits formed for each region based on both the graph and a two-layer octal coding system.
- Based on the weighting value computed with the representative ring code, the graphs are converted to a reference tree in which each node is associated with $\delta+m$ values using heuristics from several experiments, assuming each graph node has at most $m+1$ adjacent nodes.
- The vector is then generated with the first $\delta+m$ elements for the tree root and the next $\delta+m$ elements for the second tree node ranked, and so on.

This method is only suitable for block-shaped objects with rectangular view-contour boundaries. The work of Chuang *et al* [1999] provides an example.

7) Simplified skeleton

A simplified skeleton is a tree structure with line segments [Wu and Jen 1996] represented by an input vector that is formed in the following process:

- A standard tree structure in which each parent branch has the same number of descendants is predefined;
- A simplified skeleton with several standard trees is represented;

- Six attributes of a branch in the standard tree to describe each real link (non-null assignment branch) and the spatial relationships among them are defined; and
- The standard tree is converted into a vector in which each element corresponds to a branch; there can be several standard trees for a simplified skeleton.

This representation can be used to classify 3D prismatic parts, but only the contour information of the part is considered.

2.2.3 The output format

The output of an ANN is the result of many operations with the inputs and weights. Commonly, it is a nodal value in the format of a vector. Based on the information in the output vector, there are three types of output format.

1) Each neuron corresponding to a feature class

Many systems adopt this output format. In Chen and Lee's work [1998], for example, the six neurons on the output layer represent six feature classes: rectangle, slot, trapezoid, parallelogram, v-slot and triangle. Nezis and Vosniakos' system [1997] provides eight output neurons corresponding to eight feature classes.

2) Neurons representing the information of the recognised feature

Hwang [1991] uses six neurons for the output, representing the class, name, confidence factor, the main-face name, the list of associated faces of the

feature found, and the total execution time. A file storing the information of different features is constructed. In the file, each vector corresponding to a feature has thirteen elements as follows: the feature name, the number of elements in the weight vector, eight elements representing the weights, a threshold factor, a gain factor, and the number of iterations before converging to an acceptable value.

- 3) A matrix file containing the code for each recognised feature and its machining directions

The output has the information on tool access direction, which in turn reflects the feature orientation. Zulkifli and Meeran's system [1999] is a typical example: the output is a binary matrix $O=[b_{ij}]$, $1 \leq i \leq 2$, $1 \leq j \leq 5$, with b_{1j} representing the code for the feature recognised, and b_{2j} showing the tool accessibility to machine the feature, namely $+x$, $-x$, $+y$, $-y$ and $-z$ directions.

2.2.4 The training method

The training or learning method determines how the network will react when an unknown input is presented [Prabhakar and Henderson 1992]. Before the process of recognition, the neurons in the network have to be trained with some set of training features. The training process is generally classified as supervised learning or unsupervised learning. During supervised training, the correct class corresponding to the training pattern is given. The net produces an output based on its current weights, and compares it with the correct output. If there is a difference, the weights are adjusted according to a learning algorithm based on the output difference. Most ANN-based feature recognition work employs supervised

training with a back propagation algorithm [Nezis and Vosniakos 1997, Chen and Lee 1998 and Zulkifli and Meeran 1999]. During back propagation a given input, called the training input, is mapped to a specified target output. The training process comprises of four stages:

- 1) the weights are initialised;
- 2) training vectors/matrices are presented to the network;
- 3) the actual and desired outputs are compared, and the network's error is calculated as the difference between its output and target - the mean squared error is commonly used as the test norm. In the system developed by Chen and Lee [1998], the root-mean-square (RMS) error is defined by the equation:

$$RMS = \sqrt{\frac{\sum_p \sum_j (t_{jp} - a_{jp})^2}{n_p n_o}}, \quad (2-3)$$

where t_{jp} is the target value for output neuron j after presentation of pattern p

a_{jp} is the output value produced by output neuron j after presentation of pattern p ,

n_p is the number of patterns in the training set, and

n_o is the number of neurons in the output layer.

- 4) information about this error is propagated backwards to the hidden neurons and the weights adjusted accordingly.

After a number of iterations, the output will converge towards the target. The delta rule, also known as the Widrow-Hoff learning rule is used to modify the

weights [Nezis and Vosniakos 1997 and Chen and Lee 1998].

Prabhakar and Henderson [1992] developed a system that allows the feature class to be stored in the net as it is defined. Although in a sense it used a supervised training method, it is not receptive to traditional neural net training. During a training session, the trained feature is presented only once, and the weights and other parameters are set at the same time. Thus there is a lack of fundamental quality to the learning.

At present, the results of neural network-based methodologies are limited to a range of particular features which are outlined in Table 2.1. From the previous discussions, it can be seen that neural networks have the potential to devise general methods of feature recognition that are effective and robust. Most of the neural network-based systems have shown a higher recognition speed, and any features that are moderately similar to the training examples can be recognised.

Table 2.1. Capabilities of ANN-based feature recognition systems

System	Features Recognised
Prabhakar and Henderson [1992]	3-D features that can be defined by one primary face and a set of secondary faces, e.g. flat bottom hole, through-slot, through-hole
Peters [1992]	2-D features: square, rectangle, parallelogram, slot
Hwang [1991]	a. Simple and partial features whose main face must be directly connected to all its associated faces, e.g. pocket, slot, through-hole, blind hole and step b. Compound features formed by two or more non-intersecting simple features
Dagli <i>et al</i> [1993]	2-D features: bracket, circle
Chan [1994]	3-D features: block, hole, slot, pocket, groove, cylinder and boss
Gu <i>et al</i> [1995]	Depression features: step, slot, blind step, blind slot, pocket, inverted dove tail slot, blind hole
Wu and Jen [1996]	Some 3-D prismatic components
Nezis and Vosniakos [1997]	a. Features such as slot, blind slot, step, pocket and hole which only have planar faces b. Simple curved faces
Chen and Lee [1998]	2-D features: rectangle, slot, trapezoid, parallelogram, V-slot and triangle
Zulkifli and Meeran [1999]	a. Simple primitive features defined by four rectangular vertices, such as step, slot, blind slot and pocket b. Circular features c. Z-slanting features d. Non-orthogonal faces in the x and y directions
Chuang <i>et al</i> [1999]	3-D block-shaped components
Öztürk N. and Öztürk F [2001].	Non-Standard feature recognition

2.3 Design by features

In the mid-1980s feature-based design was proposed to interface CAD and CAPP [Cay and Constantin 1997]. Since then, a number of integrated CAD/CAPP systems using design by features were implemented, which can be broadly classified into two categories [Lee and Kim 1998]: Destruction by machining features, in which a design model is built by subtracting depression features from a raw stock and the machining features are derived simultaneously. Because only depression features are used, the designer's creative work is restricted; Synthesis by design features, in which a design model is generated by adding protrusion features and subtracting depression features.

2.3.1 Design by features systems

Several design by features systems have been implemented since the mid-1980s, for rotational components, prismatic components, 3D casting and sheet metals, and mechanical assemblies.

Shah and Rogers [1988] developed an expert form feature modelling shell, which supports user definition of form features. This system consists of two shells: feature modelling shell (FMDS) and feature mapping shell (FMPS). FMDS provides the necessary facilities for creating a product database except the actual definition of features, while FMPS extracts and reformulates product data as needed by the application. This work focuses on form features and is limited to 3D rotational component design.

Desai and Pande [1991] designed a feature modelling system (GFM) for CAPP for rotationally symmetric components. The modular structure of GFM consists of feature modeller, feature validator, and dimensioning and tolerancing modules.

Li *et al* [1993] presented a methodology of incorporating composite features and variational designs into a feature-based design system. Composite features are created with only a handful of primitive feature commands. Variational feature-based designs are composed of dimensional operation unit, and relative positional operator for position and feature change. This system increases the flexibility of design commands creation and the capability of variational designs, but is implemented for rotational components.

Chan and Nhieu [1993] proposed a framework for implementing a feature-based application with a CAD system, in which a user-defined external feature database was built, based on a hierarchical structure containing all feature information for the downstream application. This system is restricted to cabinet manufacturing.

Jasthi *et al* [1994] presented a methodology for developing feature-based part-modelling systems. A part description system, called TURBO-MODEL is designed to represent geometrical, technological and global data. However, only 2D rotational components can be modelled.

De Martino *et al* [1994] proposed a system consisting of a feature modeller and a feature-boundary processor. The user can design a component model directly with features defined in the common feature library. An intertwined data structure, called intermediate model is defined as a communication link between the geometric model and the feature-based model. Interacting features are

considered through the feature-boundary processor, but the result is limited. In addition, multi-mapping will result in inconsistency of information.

Perng and Chang [1997] proposed a feature-based design system for prismatic parts. The system uses high-level 3D features as basic design entities and provides a graphical-user-interface design environment with two functions: a new construction function and a dynamic editing function. The predefined 3D features considered consist of Arch, Filler, Hole, Tslot, Uslot, Vslot and Wedge. However, some technological information, such as tolerance, surface finish is not included in the feature definition.

Lee and Kim [1998] employed an incremental approach for extracting machining features from a feature-based design. Consequently, they [Lee and Kim 1999] proposed a feature-based approach to generate alternative interpretation of 3-axis milling machining features from a feature-based design model. A set of alternative generation operators, i.e., reorientation, reduction and splitting, are applied to generate alternative machining feature models. This approach uses a STEP-based feature representation scheme, but it does not integrate with design rules and constraints.

Tseng [1999] presented a modular modelling approach by strengthening the technical support provided to the designer. Because this approach creates a new component based on the existing functional modules, it cannot accommodate new components which are completely different.

Case and Harun [1999] provided a design tool to create a mechanical assembly in terms of features. A feature-based assembly modelling system is proposed and

embedded in the ACIS object-oriented solid modeller kernel. This approach shows that features can be used in multiple applications, such as process planning and assembly.

Bidarra and Bronsvort [2000] proposed a semantic feature modelling approach to defining and maintaining the semantics of feature during all the modelling operations. Feature class specification, feature model structure and functionality, and model validity maintenance scheme are included. The semantic feature modelling is defined by high-level programming languages, such as object-oriented languages, and therefore contains fewer errors. However, the imposing rigid validity rules reduce the modelling freedom of the user.

Other work includes Turner and Anderson [1988], Chang [1989], Anderson and Chang [1990], Pratt [1990], and Chung *et al* [1990].

2.3.2 Design by features techniques

Two important techniques for design by features have been reported: feature representation and feature validation.

1) Feature representation

Feature representation including feature definition and data structure, is an important aspect for design by features. Kang and Nnaji [1993] presented a generalised feature definition, classification and representation in the domains of mechanical assembly and sheet metal fabrication. A face-based feature graph was proposed to represent features by a face-face adjacency graph and face-edge incidence matrix in their research. Hounsell and Case [1999]

applied a method to identifying structured spatial geometric feature interactions based on a broad multilevel classification. Simple Boolean expressions are used to identify the relationship between any two entities, such as disconnected entities and connected entities. Wu *et al* [2001] proposed a face-based mechanism for naming, recording and retrieving topological entities to replay the design history.

2) Feature validation

Feature validation is another important issue addressed in design by features. Gindy *et al* [1998] proposed a Boolean operation-based systematic methodology of automating feature validation. The feature validation concerns with feature interactions, feature dimensions and feature class. Three procedures of feature validation are includes; they are adding a new feature, deleting an existing feature and modifying an existing feature. Case and Hounsell [2000] presented a methodology to validate the feature-based representation by capturing and using the designer's intents related to functional, relational and volumetric aspects of the component geometry.

In contrast to feature recognition, design by features can capture the design and manufacturing information during the design stage. It reduces remarkably the amount of work for recognising features, but does not eliminate the need for feature recognition.

In summary, previous research on design by features have gained some achievements in system development, feature definition, data structure of modelling and feature validation. However, several problems need to be tackled:

- 1) Feature interactions are yet to be fully resolved.
- 2) Standard feature classification and universal feature taxonomy are needed.
- 3) Enhancement is necessary in feature validation.

2.4 Neural network-based CAPP

Process planning is a function that establishes a set of manufacturing operations and their sequence, and specifies the appropriate tools (machine tools, cutting tools, jigs and fixtures, etc.) and process parameters in order to convert a part from its initial raw state to a final form predetermined from an engineering drawing [Devireddy and Ghosh 1999]. In the past, process plans were often generated by human process planners, which led to inconsistency, low efficiency and slow response to the changes in design and production environment. In recent decades, with the advents of computer technologies and artificial intelligence (AI) techniques, it has become easier to undertake process planning on the computer, that is, computer-aided process planning (CAPP). The use of computer techniques to automate the tasks of process planning has been the subject of extensive research. Although there are a huge amount CAPP system have been reported over the past two decades, the results are still far from being a practical industrial application.

In recent years, ANNs have offered an encouraging approach to CAPP because of their learning ability. This section details the ANN techniques used in CAPP in

the following areas: the network topology, input representation, output representation, training method and a discussion of achievements.

2.4.1 The topology of artificial neural network

To date, four main ANN architectures have been used in CAPP: Feed-Forward network (FF), Hopfield network, Brain-State-in-a-Box (BSB) and MAXNET.

1) Feed-Forward network (FF)

Most neural network-based CAPP systems use the feed-forward architecture [e.g. Li *et al* 1994]. In general, the appropriate structure is identified through several experiments during the learning process. The structure with the minimum errors and the fastest learning rate is chosen. According to the number of hidden layers, the net topology can be classified into three architectures.

- Three-layer feed-forward network

As described in Section 1, it comprises the input, hidden and output layers. This structure is suitable for a mapping in a continuous decision region [Mei *et al* 1995]. Osakada and Yang [1991] related the shape of rotationally symmetric products to their forming methods. The network consists of a 256-unit input layer, an 8-unit hidden layer and a 4-unit output layer. Gu *et al* [1997] employed a three-layer feed-forward network with a 5-neuron hidden layer for manufacturing evaluation. Santochi and Dini [1996] proved in their experiment that a three-layer feed-forward network with a suitable number of

neurons for each layer is the best architecture for selecting technological parameters for a cutting tool using the hyperbolic tangent sigmoid function.

- Four-layer feed-forward network with two hidden layers

Park *et al* [1996] developed a 4-layer neural network to modify cutting condition based on several tests. Their network has a 15-neuron input layer, two 15-neuron hidden layers and a single-neuron output layer. Although four-layer feed-forward networks are more versatile than three-layer feed-forward networks, they train more slowly due to the attenuation of errors through the non-linearities [Principe *et al* 2000].

- Five-layer feed-forward network

Le Tumelin *et al* [1995] proposed a 5-layer feed-forward network to determine appropriate sequence of operations for machining holes.

2) Hopfield network

The Hopfield network is a single layer recurrent network that uses threshold process elements and an interconnect symmetric matrix as shown in Figure 2.3 [Principe *et al* 2000]. A minimum point or attractor has been demonstrated to be existence in this network, which corresponds to one of the stored patterns. It can be described as the following [Principe *et al* 2000]:

$$y(n+1) = \text{sgn}\left(\sum_{j=1}^N w_{ij}y_j(n) - b_i + x_i(n)\right) \quad (2-4)$$

where $i=1, \dots, N$,

sgn represents the threshold nonlinearity $(-1, 1)$, and

b_i is a bias.

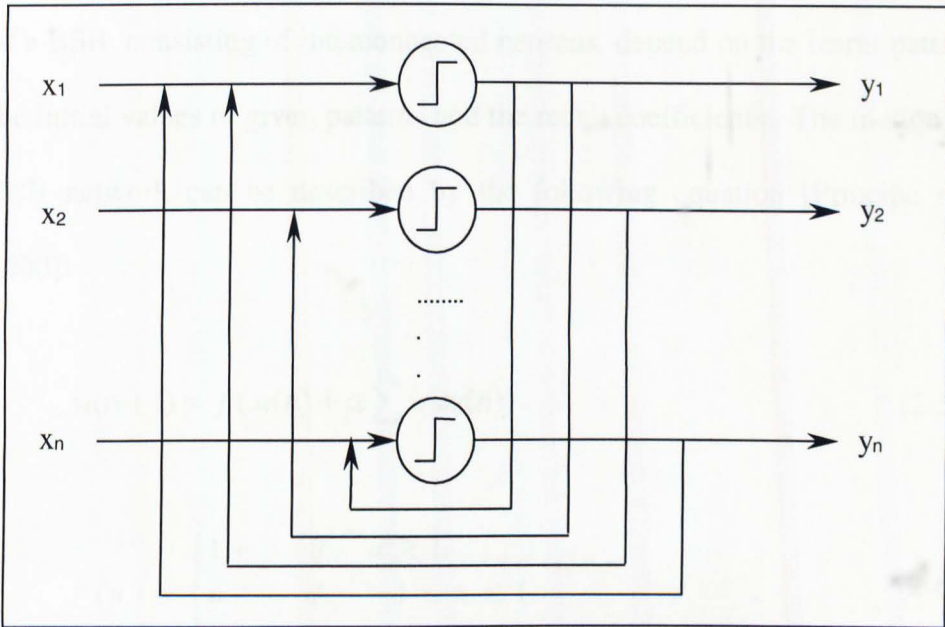


Figure 2.3 Example of a Hopfield network

The dynamics of the Hopfield network can be described by the state of an energy function which eventually gets to a minimum point. Therefore, optimal operation sequencing can be expected with the continuous downward trend of a global energy function. Shan *et al* [1992] adapted the Hopfield network to the operation sequencing problem. Supposing the number of operations is n , the network is then composed of n^2 neurons, each identified by double subscripts: the operation and the sequence to be executed.

The Hopfield network provides one of the strongest links between information processing and dynamics. However, spurious memories limit its capacity to store patterns.

3) Brain-State-in-a-Box (BSB)

As a discrete-time recurrent network with a continuous state, the output values of a BSB, consisting of interconnected neurons, depend on the learnt patterns, the initial values of given patterns and the recall coefficients. The motion of a BSB network can be described by the following equation [Principe *et al* 2000]:

$$y_i(n+1) = f(x_i(n) + \alpha \sum_{j=1}^N w_{ij}x_j(n)) \quad (2-5)$$

$$f(u) = \begin{cases} 1 & \text{if } u \geq 1 \\ u & \text{if } -1 \leq u \leq 1 \\ -1 & \text{if } u \leq -1 \end{cases} \quad (2-6)$$

A BSB can be used as a subnet for decision feedback applications because it amplifies the present input until all neurons saturate, and eventually converges to one of the corners of the hypercube $[-1,1]^n$.

Sakakura and Inasaki [1992] adapted a BSB network in a CAPP system. The number of neurons assigned for the dressing depth of cut, dressing feed and surface roughness are 5, 5 and 9 respectively. The initial values are given by a feed-forward network run at the same time. The BSB repeats, performing a calculation using the following equation until the output value of each neuron converges to a certain value:

$$o_m = \text{LIMIT}(a_m) \quad (2-7)$$

$$a_m = c_1 \sum_{n \neq m} w_{mn} o_n + c_2 o_m + c_3 o_m(0) \quad (2-8)$$

where $\text{LIMIT}()$ is the function which limits the value in the parentheses between -1.0 to +1.0;

c_1, c_2, c_3 are recall coefficients; and

$o_m(0)$ is initial value of neuron m .

The limitation of the BSB network is that the location of the attractors must be predefined as the vertices of the hypercube.

4) MAXNET

MAXNET is a competitive network in which only one neuron will have a non-zero output when the competition is completed. The network consists of interconnected neurons and symmetric weights. There is no training algorithm for MAXNET and the weights are fixed as depicted in Figure 2.4 [Fausett 1994]. Its application procedure includes two steps:

- activation and initialisation of weights, and
- updating the activation of each unit until only one unit responds.

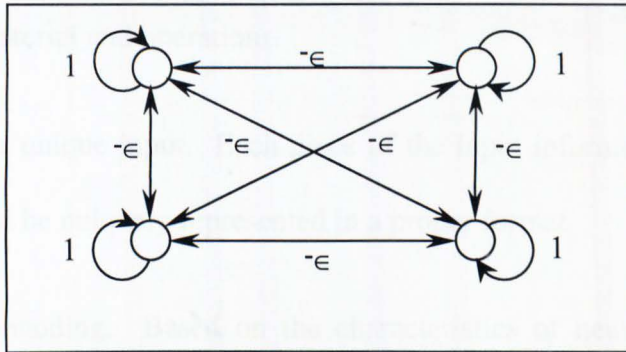


Figure 2.4 Example of MAXNET

MAXNET is suitable for situations where more information is needed than can be incorporated. Knapp and Wang [1992] used a MAXNET to force a decision between the competing operation alternatives. In their work, a sequence of operations for machining each feature of the part is generated independently by the MAXNET.

However, few systems use only BSB or MAXNET independently. BSB and MAXNET are typically used in multi-type architectures, e.g. Sakakura and Inasaki [1992] used a BSB with a three-layer feed-forward network while Knapp and Wang [1992] utilised a co-operating architecture combining a three-layer feed-forward network and a MAXNET.

2.4.2 Input representation

The input representation for neural network-based CAPP involves the conversion of design data into a proper input format. There are three aspects to be resolved.

- 1) Input information. Process planning deals with a number of detailed activities, such as selecting manufacturing operations, determining setups, specifying appropriate tools and so on. Each activity requires a different set of information. For example, setup generation requires information about

tolerance, material and operations.

- 2) The need for unique input. Each piece of the input information for a neural network must be uniquely represented in a proper format.
- 3) Numerical encoding. Based on the characteristics of neural networks, the input parameters need to be converted into numerical values.

There are four main types of input representation.

- 1) Standardised image data

Osakada and Yang [1991] converted the cross-sectional shape data of the product into standardised image data for the input. They use 12 "colours" to represent 12 outer or inner geometric primitives, such as cylinder and cone. Half of the product shapes are converted into a 16*16 "colour" data image. These 256 units are regarded as the input to the neural network. This representation can only be used for rotationally symmetric products.

- 2) Input vector with value ranging from 0 to 1

It is one of n -unit vector input formats, whose units are coded with numerical values ranging between 0 and 1. Certain special transformations have to be performed, which different formulae need to be established for different units. For instance, a unit related to the workpiece material is calculated from the cutting force per unit area, k_c using the following formula [Santochi and Dini's work, 1996]:

$$i_6 = \frac{k_c - 1900}{2600} \quad (2-9)$$

Park *et al* [2000] defined the input values from 0 to 1 according to its real values, e.g. 0.5000 for a hardness unit for a real value of 225 BHN, and 0.4366 for a cutting speed unit for a real value of 80 m/min.

3) Input vector with integer value

Each input unit is given a particular integer instead of the original value. A common method is to classify the value for each parameter and assign a discrete integer to the corresponding unit. Park *et al* [1996] used 15 input parameters concerning seven factors, such as the feature type, ratio of feature width to depth, tool length, tool material. A class number is given for each parameter based on its real value, e.g. the class number is 6 if the ratio of selected tool length over standard length is 2. Similarly, in the system by Sakakura and Inasaki [1992], the unit of dressing depth of cut is assigned a value of 4 for a real value of 11.0 μm . Mei *et al* [1995] developed a scheme for rotational parts in which the surface orientations (i.e. right, left or both) are represented by values -1, 1 and 0.

4) Input vector in binary form

The value of each unit uses only two characters (i.e. 0 and 1) representing whether the corresponding parameter is needed or not. In order to determine feature clusters, Chen and LeClair [1993] represented a feature with a $(6+n)$ -

unit vector in binary form, which defines the *six* approach directions and *n* tool types.

2.4.3 Output representation

Five main output formats have been proposed which are summarised below.

1) Output vector in ordered binary form

The output vector consists of a number of neurons, each with a value (i.e. 0 or 1) showing whether the corresponding item (e.g. machining operation, tool and so on) belongs to the process plan or not. The output vector in the first stage consists of eight neurons representing respectively drilling, reaming, boring, turning, taper turning, grooving, grinding and precision. If the value of a neuron is '1', the corresponding operation is needed for the feature; otherwise, the value is '0'. For instance, a hole requires the drilling operation, so the first neuron is assigned the value '1'. The sequence of the vector represents the sequence of the operations, e.g. reaming is usually performed after drilling. Li *et al* [1994] used a 4-neuron vector corresponding to the abrasive type, grade, grit size and bond. Le Tumelin *et al* [1995] designed a 23-neuron vector.

2) Output vector with special values

Each neuron in the output vector has a possible value that the corresponding parameter may assume. Santochi and Dini [1996] developed a system for selecting the eight technological parameters of a cutting tool. For example, to select a normal clearance angle α_n , the number of output neurons is 5 which

represents 4° , 5° , 6° , 7° , 8° respectively. The neuron with the value '1' represents the optimal value and '0.5' a second choice.

3) One-unit output in binary form

This output format has only one unit whose value is either 0 or 1 [Mei *et al* 1995]. The output shows which surfaces should be used as manufacturing datums. For instance, '1' means that the surface will be used for the part setups and '0' means that it has nothing to do with part setups.

4) One-unit output in integer form

Each discrete integer is concerned with a special class [Chen and LeClair 1993]. The output integer represents a cluster of features according to the approach directions and the tool types.

5) Output matrix

Shan *et al* [1992] devised a binary incidence matrix V ($n*n$) in which the rows denote operations and the columns correspond to sequences. The value '1' indicates that a specified operation is performed. Because each operation is performed only once and only one operation is carried out at a time, one and only one of the entries in each row and column should take the value of 1 whereas the rest should be set to 0.

2.4.4 Training method

In CAPP applications, the training method usually employs either an unsupervised learning algorithm or back-propagation.

1) Unsupervised learning algorithms

With an unsupervised learning algorithm, the training set only contains input samples; no desired or sample outputs are available. The neural network must construct an internal model that captures regularities in input training patterns instead of measuring its predictive performance for a given input. Hence this method is also called self-organisation. In CAPP applications, a logical AND/OR operation-based unsupervised learning approach is used. Chen and LeClair [1993] clustered features based on the approach direction and tool type and then generated a process plan using an Episodal Associative Memory (EAM) approach. The AND operation was applied to solve multiple approach directions for some features. If the digit is 1 for the corresponding approach direction, the update weight for the cluster j is

$${}^a b_{ij}(s+1) = {}^a x_i^{(p)} \text{ AND } {}^a b_{ij}(s) = {}^a x_i^{(p)} {}^a b_{ij}(s) \quad (2-10)$$

where ${}^a x_i^{(p)}$ is the approach direction sub-pattern, $\langle +x, +y, +z, -x, -y, -z \rangle$, of pattern p .

In the meantime, the OR rule is used to update the weight so that the probability of common tools can be increased. If the digit is 1 for the corresponding tool, then $b_{ij}(s+1)$ is modified according to the following equation [Chen and LeClair 1993]:

$${}^t b_{ij}(s+1) = {}^t x_i^{(p)} \text{ OR } {}^t b_{ij}(s) = f({}^t x_i^{(p)} + {}^t b_{ij}(s)) \quad (2-11)$$

where ${}^t x_i^{(p)}$ is the tool sub-pattern, and

$$f(\eta) = 1 \text{ if } \eta \geq 1, \text{ else } f(\eta) = 0.$$

2) Back-propagation

A back-propagation algorithm is a form of supervised learning. The algorithm consists of two basic steps:

- initialisation of weights. In Gu *et al's* work [1997] all the weights were initially randomly set in the range 0 to 0.1;
- repetition of training until the error is acceptably low. Gu *et al* [1997] mapped the selected pattern pairs to reinforce the weights until the deviation between the training output and the target output of each sample converged to a pre-defined error goal (e.g. 0.05).

Back-propagation methods have proven highly successful in CAPP applications [Osakada and Yang 1991, Dong *et al* 1995 and Mei *et al* 1995]. These method can be classified into three groups.

- The delta rule

One of the back-propagation learning algorithms is the delta rule based on the cumulative error. It is also known as the least mean squares (LMS) or Windrow-Hoff rule. The learning rule changes the connection weights so as to minimise the mean squared error between the network output and the target over all training patterns.

Sakakura and Inasaki [1992] chose the delta rule for both a feed-forward network and a BSB network. In the three-layer feed-forward network, the weight connecting neuron j in the hidden layer to neuron k in the output

layer is updated as follows:

$$\Delta W_{kj} = \eta_f \sum_p \delta_{pk} o_{pj} \quad (2-12)$$

$$\delta_{pk} = (t_{pk} - o_{pk}) f'(a_{pk}) \quad (2-13)$$

where $f()$ is the output function of neuron,

η_f is the learning coefficient of the FF network,

p is the learning pattern number,

t_{pi} is the learning value of neuron i for learning pattern p ,

a_{pi} is the status value of neuron i for learning pattern p , and

o_{pi} is the output value of neuron i for learning pattern p .

For the BSB network, the modified value of the weight which interconnects neuron m and neuron n is calculated as the following:

$$\Delta W_{mn} = \eta_b \sum_p (t_{pm} - w_{nm} t_{pn}) t_{pn} \quad (2-14)$$

where η_b is the learning coefficient of BSB network, and

t_{ji} is the learning value of neuron i for learning pattern j .

- Levenberg-Marquardt approximation

A back-propagation algorithm using the approximation of Levenberg-Marquardt is also used in some applications [Santochi and Dini 1996]. This algorithm allows a better performance in terms of training time in comparison with other training methods. However, it may require a very

large storage space for some complex situations.

The matrix of the connection weights is updated through the following equation:

$$w = (J^T J + \mu U)^{-1} J^T e \quad (2-15)$$

where J is the Jacobian matrix of derivatives of the errors to each

weight $w_{i,j}$,

μ is a scalar,

U is the unit matrix, and

e is the error vector of the network.

- Batch training

Either the delta rule or the Levenberg-Marquardt approximation is used as the on-line learning rule. The batch training is an off-line training process. Rather than adjust the weights after each pattern presentation, batch training accumulates the errors over the whole training set and adjusts each weight according to the accumulated errors. It can generally be expressed as follows [Principe *et al* 2000]:

$$\Delta w_{ji} = \eta \sum_p \delta_{out} H_{in} \quad (2-16)$$

where the subscripts *in* and *out* refer to the net input and output signals associated with a given unit, and

i and j refer to the connection from unit i to unit j .

The form of δ will vary depending on the type of layer to which the formula applies. In some cases it is advantageous because of its smoothing effect on the correction terms and increasing of convergence to a local minimum [Fausett 1994]. Devireddy and Ghosh [1999] trained a system with a batch training back-propagation algorithm.

The achievements of neural network-based CAPP systems are summarised in Table 2.2. The major drawbacks of the existing neural network-based CAPP systems are:

- 1) Lack of systematic investigation of the framework and methodology of CAPP. Most of solutions are designed for specific activities (e.g. tool parameters selection, cutting condition generation) and specific applications (e.g. cold forging, grinding operations), which cannot be used in different industrial environments.
- 2) Low efficiency and quality of process planning. For a process planning system based on the machined surface, with the number of machined surface increasing, the efficiency and quality of reasoning can not guarantee, especially for a complex component.
- 3) There have not been so many systems considering prismatic components. It is mainly due to the complex geometrical representation of the 3D prismatic components and the intricate nature of cutting mechanism in milling. It is difficult for a CAPP system to plan a solution for all possible components.

Although the capabilities are limited at present, there is great potential for the

application of neural networks to CAPP. The review has shown that neural network techniques can significantly improve the performance of CAPP systems. The self-learning functions allow empirical rules to be learnt through typical examples. Faster processing makes systems more effective, especially in parallel environments.

Table 2.2. Achievements of ANN-based CAPP systems

System	Functions	NN type	Manufacturing processes / components
Osakada and Yang [1991]	Generation of process plan	Three-layer Feed-Forward network	Cold forging for axis-symmetric components
Roy <i>et al</i> [1994]	Generation of process plan	Feed-Forward network	Cold forging
Knapp and Wang [1992]	Operation selection and operation sequencing	Three-layer Feed-Forward network and MAXNET	Machining operations
Devireddy and Ghosh [1999]	Operation selection and operation sequencing	Three-Layer Feed-Forward network	Machining operations for rotational components
Shan <i>et al</i> [1992]	Operation sequencing	Hopfield	Cutting operations for components machined on single spindle Swiss-type automatics
Le Tumelin <i>et al</i> [1995]	Operation sequencing	Five-layer Feed-Forward network	Machining operations for holes
Dong <i>et al</i> [1995]	Operation sequencing	Feed-Forward network and Hopfield	Machining operations
Gu <i>et al</i> [1997]	Operation sequencing	Three-layer Feed-Forward network	Machining operations for prismatic components with regular machining features
Giusti <i>et al</i> [1986]	Tool selection	Unknown	Rotational components

Chen and LeClair [1993]	Generation of setups	Unknown	Machining operations
Santochi and Dini [1996]	Selection of optimal values of a tool parameter	Three-layer Feed-Forward network	Turning operations
Mei <i>et al</i> [1995]	Selection of manufacturing datums	Three-layer Feed-Forward network	Rotational components
Li <i>et al</i> [1994]	Selection of grinding wheels	Feed-Forward network	Grinding operations for ground components
Sakakura and Inasaki [1992]	Selection of dressing conditions	BSB/Three-layer Feed-Forward network	Grinding operations for ground components
Park <i>et al</i> [1996]	Generation of modified cutting conditions	Four-layer Feed-Forward network	Milling and turning for sheet metal
Park <i>et al</i> [2000]	Generation of cutting conditions	Fuzzy ARTMAP network	Milling operations

2.5 CAPP using genetic algorithms

Operation sequencing is a task responsible for arranging the selected operations in a suitable order to fabricate the part [Shan *et al*, 1992]. An optimal process sequence could largely increase the efficiency and decrease the cost of production. Therefore, operation sequencing is always the major concern among process planning activities. It is influenced by various constraints and factors, such as the selection of machining operations, the machine tool chosen, tools accessibility,

geometrical tolerance and manufacturing rules. Thus, the task of sequencing is a problem to consider different choices and constraints. Genetic algorithms (GAs) are a technique for seeking to 'breed' good solutions to complex problems by survival of the fittest. They have been successfully applied to various optimisation problems since the mid-1960s. Some attempts using GAs have been made on operations sequencing optimisation, .

Vancza and Markus [1991] applied a genetic algorithm to optimising sequence of operations for machining a component composed of milling features, i.e. slots and holes. In their system, each string is represented by elements corresponding to feature states that are produced by machining operations. Three factors are considered in their optimisation criteria, that is, the number of setups of each plan, the total number of tool changes, and the total cost of individual operations.

Yip-Hoi and Dutta [1996] used a genetic algorithm to generate plans for machining a milled /turned component for parallel machining that satisfied both the constraints of the geometry of the component and the restriction due to the configuration of the machine-tool environment. A new coding method that allows the generation of only valid operation strings is developed, but operation features with multiple parents are not considered. The planning criterion they used is to minimise the component processing time.

Dereli and Filiz [1999] developed an optimisation system for process planning using genetic algorithms. In their research, a reward/penalty matrix called REPMAX for each setup is determined based on the selected criterion, such as

safety or minimum tool change, and the objective of optimisation is to gain the least total penalty or largest total reward.

Bhaskara *et al* [1999] proposed a quick identification of optimal or near optimal operation sequences in a dynamic planning environment using a genetic algorithm. They identified the feasible sequences based on a Feature Precedence Graph and used minimum production cost as the objective function, which was calculated from a precedence cost matrix. The precedence cost matrix is generated for any pair of features based on the relative costs corresponding to the number of tasks that need to be performed in each category of attributes such as machining parameter change, tool change, setup change, machine change and the type of constraint one feature has with the other.

Qiao *et al* [2000] presented a genetic algorithm method to select the machining operation sequence for prismatic parts. A combination fitness expression F is defined as the following:

$$F = W_p F_p + W_c F_c + W_{ao} F_{ao} + W_{op} F_{op} \quad (2-17)$$

where F_p is the fitness of precedence rules;

F_c is the fitness of clustering rules;

F_{ao} is the fitness of adjacent order rule;

F_{op} is the fitness of optimisation objectives (In the practical system developed by Qiao *et al* [2000], F_{op} is not considered); and

W_p, W_c, W_{ao}, W_{op} are the weights of F_p, F_c, F_{ao}, F_{op} , respectively.

In addition, some biological parameters are discussed, including the number of individuals, the number of parent individuals and the mutation ratio.

Li *et al* [2002] developed a hybrid genetic algorithm and simulated annealing approach for optimising process plans for prismatic components. A modified fitness function FF is defined as:

$$FF = \lambda * (UL - TWC) * (UL - TWC) \quad (2-18)$$

where UL is an upper limit constant for TWC ;

TWC is the total weighted cost;

λ is a positive coefficient.

In addition, Shunmugam *et al* [2000] and Dereli *et al* [2001] used genetic algorithm for selecting optimal machining parameters, respectively. Li *et al* [2002] also designed a constraint adjustment algorithm to rearrange the process plan according to the constraints while some random properties in it are kept.

In conclusion, GAs are quite promising in identifying optimal operation sequences. The effectiveness of a GA depends on the fitness function, appropriate crossover and mutation operators. The problems with existing GA-based methods for process planning include

- 1) Few systems have intended to provide a globally-optimised fitness function definition. As we know, operation sequencing is a comprehensive problem concerning various factors, such as processing cost, processing time and manufacturing rules. Although there have been several fitness functions for

process sequencing, e.g. minimum number of setups, minimum machining cost, shortest processing time or satisfaction of manufacturing rules, etc., they are used only for local optimisation for the chosen object.

- 2) Lack of adaptability. Most of GA-based CAPP systems proposed up to date are designed for specific industrial environment and it is difficult to adapt them for different industrial applications.
- 3) Inability to learn. GA-based CAPP systems cannot learn to adapt themselves automatically from the external environment.

From the discussions, it still can be seen prospectively that GAs can be further applied to process planning, e.g. incorporating other techniques such as ANNs for global optimisation.

2.6 Summary

This chapter has presented a state of the art review of three key research issues in CAD/CAM integration: feature recognition, design by features and CAPP. The research reviewed above demonstrates that

- 1) There are still some problems to be solved with current methods, such as poor ability to recognise intersecting features and adaptability to CAPP. Therefore, new methods to recognise intersecting features and adapt CAPP are needed;
- 2) There are potential benefits for using neural networks in feature recognition with a design by features approach. Input representation of features for neural network-based feature recognition is a main problem that needs to be solved.

3) A hybrid GA method with neural network and fuzzy logic in CAPP is applicable.

Therefore, a CAD/CAM integration methodology using a neural network, GA and fuzzy logic is presented in this thesis in order to overcome the limitations in current methods.

Chapter 3.

Design by Features Module

As mentioned in Chapter 2, to take advantages of both design by features and feature recognition, the current research has employed both approaches, incorporating a new identification algorithm of interacting features with ANN techniques. The ANN-based feature recogniser is used for the identification of new classes from interacting features while design by features techniques keep record of all feature information. This chapter will discuss the proposed design by features module. Chapters 4 and 5 will detail the identification algorithm for interacting features and ANN-based feature recogniser.

3.1 Architecture of the design by features module

Taking into account the advantages and disadvantages of design by features (see Section 2.3) and the requirements for integrating feature recognition, an architectural framework for a design by features module has been proposed in terms of certain fundamental design decisions that influence the overall system performance. As shown in Figure 3.1, the proposed architecture includes a standard feature library, feature library management, feature-based model and feature-based model management. Important characteristics are described below.

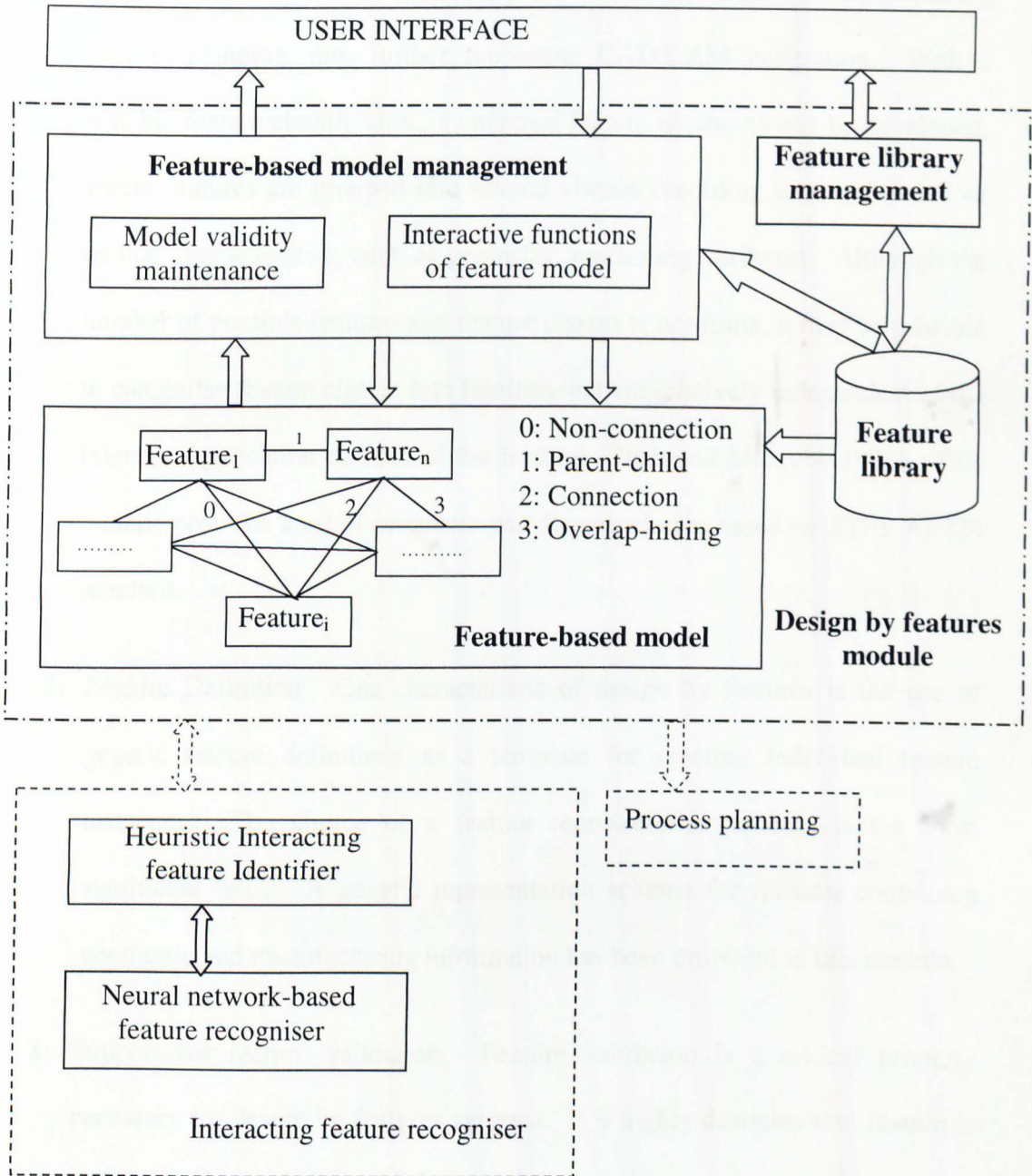


Figure 3.1 Architecture of the design-by-features module

- 1) Feature classification: The aim of feature classification is to acquire and process similar geometric and technological data for design and manufacturing process planning, thus further supporting CAD/CAM integration. With a suitable feature classification, a universal feature taxonomy can be developed, where features are grouped into several classes according to some shared or similar characteristics, such as geometry, machining attributes. Although the number of possible features and feature classes is not finite, it may be possible to categorise feature classes into families that are relatively independent of the intended application domain of the features [Shah and Mäntylä, 1995]. This module provides a set of prismatic-part feature classes based on STEP AP224 standard.
- 2) Feature Definition: One characteristic of design by features is the use of generic feature definitions as a template for creating individual feature instances. The choice of a feature representation scheme is the most significant issue. A genetic representation scheme for features containing geometric and manufacturing information has been proposed in this module.
- 3) Support for feature validation: Feature validation is a critical property necessary for design by features systems. It is highly desirable that feature is validated when it is placed into the model and that the validity is maintained afterwards. In this research, pre-designed validity constraints are applied during the feature instancing stage.
- 4) Feature-based model: The basic entity in feature-based model is feature. Two main problems that a feature-based model has to be solved are:

- Definition of feature semantics and the relationships among features.
- Maintenance of the feature-based model.

Data structure definition and feature-based model management provide solutions to these problems.

3.2 Feature classification

There are a number of feature classification schemes. Among them are those based on geometrical properties of the features, such as the work of Gindy [1989]. Others are based on machining methods associated with features that include rotational features created by machining operations on a lathe or a turning machine, and prismatic features created by machining operations on a milling machine or a three-axis machining centre [Tseng and Joshi, 1998]. There also are those based on the number of possible tool approach directions that can be used to machine them: STAD (single tool axis direction) and MTAD (multiple tool axis direction) [Chu and Gadh, 1996]. These classification schemes have advantages in certain respects, but major problems (e.g. non-standard and incompleteness) hinder their practical applications in integrated environments for design and manufacturing.

An important feature classification different from above mentioned is provided by STEP (STandard for External representation of Product data) [STEP, 1999]. Some researchers have developed feature recognition methods based on STEP. For example, Bhandarkar and Nagi [2000] developed a Boundary-representation

(B-rep) based feature extraction system that takes a STEP file as input and produces a form-feature STEP file; and Han *et al* [2001] proposed a geometric reasoning feature recognition kernel using STEP as input and output formats. In ISO 10303 STEP-AP224 (Mechanical product definition for process planning using form features), machining features are defined as a type of manufacturing feature that identifies a volume of material to be removed to obtain the final geometry from the initial stock [STEP, 1999]. Sixteen machining feature classes are defined, such as hole, slot, etc. As an international standard, STEP has some advantages of feature classification. For example, the feature definition is universal and includes both geometric and manufacturing information. However, there are still some limitations discussed below.

- 1) The classification is not rigorous. Some overlapping or duplication situations exist. For example, the following definitions are provided by STEP.

Rectangular-open-pocket: A *rectangular-open-pocket* is a type of *pocket* that is an open profile with opposite sides that are of equal length and with one side that does not make contact with the part. A *pocket* is a type of machining-feature that is a volume with a specific shape, removed from the part.

Flat-slot-end-type: A *flat-slot-end-type* is a type of *slot-end-type* that is an end condition of a slot that shall be a planar shape perpendicular to both of the adjacent slot wall surface. The intersection of the slot wall surfaces and the end planar shape need not be blended by a radius. A *slot* is a type of

machining-feature that is a channel or depression with continuous direction of travel.

Based on the above definitions, the entity shown in Figure 3.2 can be regarded as either a blind slot with a flat-slot-end-type or a rectangular-open-pocket. Since feature overlap or duplication can make feature recognition more difficult and feature libraries larger and unmanageable, it is desirable to reconcile and remove such overlapped distinctions between features.

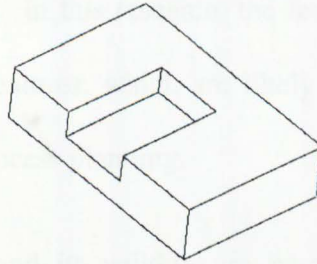


Figure 3.2 Example 1 of STEP classification

- 2) The classification is incomplete. Completeness is an important characteristic of classification, which implies that the defined feature classes are sufficient for creating models that the system is intended. In an integrated design and manufacturing system, it should include all primitive machining features. But STEP does not cover the full range, e.g. the feature shown in Figure 3.3.

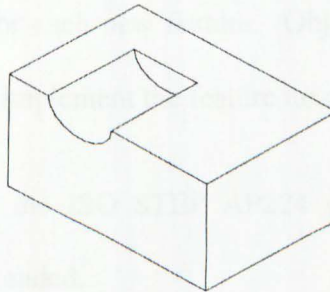


Figure 3.3 Example 2 of STEP classification

- 3) The definition of machining features is not precise. STEP defines machining features as a volume of removal material while protrusion features, which are not removed volumes, are included.

In order to overcome these limitations, a hierarchical classification (see Table 3.1) is proposed based on the following principles.

- 1) A machining feature is defined as a geometrical entity, which is related to a group of particular machining processes and can be used to reason about a suitable machining method. In this research, the features considered cover a majority of the primitive features, which are likely to be of interest for the application of machining process planning.
- 2) The feature classification and its validity are based on a multi-viewpoint considering manufacturing requirements with topological information.
- 3) If a set of features have the similar geometric and topological characteristics and can be machined with the similar process, they are called a feature class. A sub-class is regarded as an instance of its main class.
- 4) The classification is hierarchical, where a subclass inherits common properties from a higher class. This reduces the number of properties that have to be independently specified for each new feature. Object-oriented programming language - C++ is used to implement the feature taxonomy.
- 5) The feature definition in the ISO STEP AP224 standard is considered as guideline for industrial extended.

Table 3.1 Classification of machining features

Internal feature	Round hole	Through round hole	
		Blind round hole	Flat
			Flat_with_radius
			Spherical
	Conical hole		
		General hole	
	Slot	Through slot	
		Blind slot	Flat
			Woodraff
	Pocket	Closed pocket	
		Open pocket	
	Step	Through step	
		Blind step	
External feature	Revolved-feature	General_revolution	
		Groove	
		Revolved_flat	
		Revolved_round	
	Boss	Cylinder boss	
		Conical boss	
	General_outside	Close	
		Open	
	Round_end		
	Spherical_cap		
	Protrusion		
Surface machining			
Attaching feature	Knurl		
	Thread		
	Marking		
Compound feature	(not applicable in this work)		

The classification has some advantages as listed below.

- 1) With a finite feature library, it includes the majority of the primitive features likely to be of interest for the application of machining process planning.
- 2) It is more suitable for integrated CAD/CAM environments because the requirements of both design and manufacturing are considered.
- 3) It enables the use of a code representation and a computationally efficient feature recognition strategy.

The feature definition in the taxonomy is described in Appendix A.

3.3 Feature class definition

3.3.1 The requirements of feature class definition

In order to ensure a valid component model and suitable data for future process planning, feature definition must satisfy the following requirements.

- 1) Completeness: Completeness implies that a component C can be fully produced by a set of machining features $F = \{ft_1, ft_2, \dots, ft_n\}$, that is

$$C = S - \bigcup_{ft_i \in F} ft_i \quad (3-1)$$

where S represents the stock.

- 2) Connection: Each feature, ft_i should have at least one face connecting with C , that is

$$(ft_i \cap C) = \text{Face/Faces} \neq \emptyset \quad (3-2)$$

and

$$(ft_i \cap (S-C)) = \text{volume} \neq \emptyset \quad (3-3)$$

- 3) Comprehensiveness: In order to pave the way for further applications (e.g. process planning), feature definition needs to store both geometric and process data relevant to the application. Several types of attributes and the permissible attribute values for feature instantiation must be determined: geometric, manufacturing data and validation constraints.

3.3.2 The parameters of feature class definition

The parameters in feature definition are used for determining not only how a feature class is presented to the user but also how the user interacts with the system. A set of parameters of a feature class is defined based on the requirements for both design and machining purposes. The feature class in the module, which is either additive or subtractive, is represented by the feature volume or its boundary elements as a whole. As illustrated in Figure 3.4, a standard feature class can be explicitly defined with five describers: identifier, interface parameters, validity constraints, machining attributes and UndiGraph.

- 1) Identifier: A number of basic terms understandable to both the designers and the system, namely, feature name, feature code and feature class type.

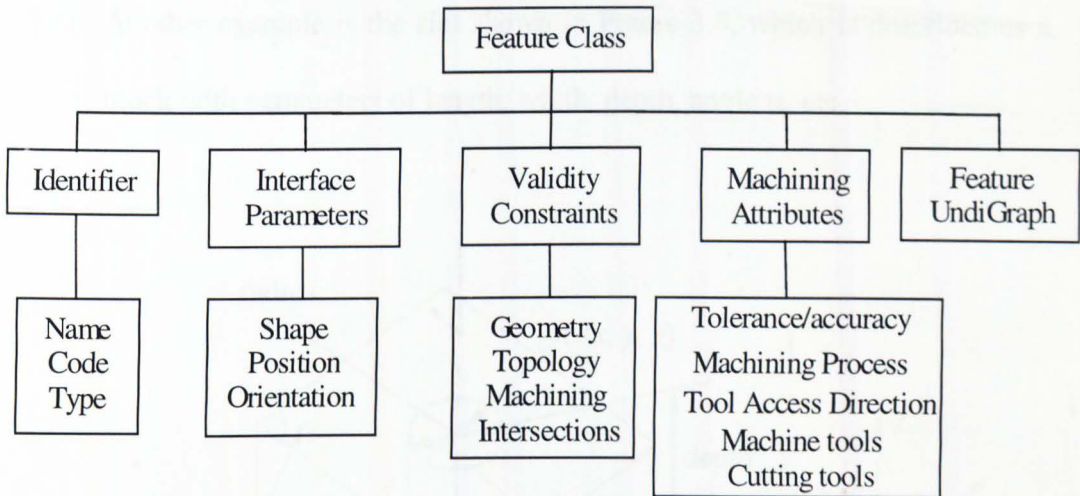


Figure 3.4 Definition of feature class

2) Interface parameters: A set of parameters relating to its basic shape and spatial relationships to the world co-ordinate system by fixing its degrees of freedom. It provides an interface between the design module and the user. These parameters will be specified at the time that the feature is created. There are two major categories.

- Geometric shape

The basis of a feature is its geometric shape, which indicates the volume of material added to or removed from the model. In the proposed design by features method, geometric shapes of features are determined directly by specifying geometric shape parameters predefined in the feature class templates. The parameter set must be sufficient to construct but not over constraint a feature. For example, two geometric parameters of feature hole, radius and depth, constitute its basic shape - a cylinder (Figure 3.5).

Another example is the slot shown in Figure 3.6, which is described as a block with parameters of length, width, depth, angle α , etc.

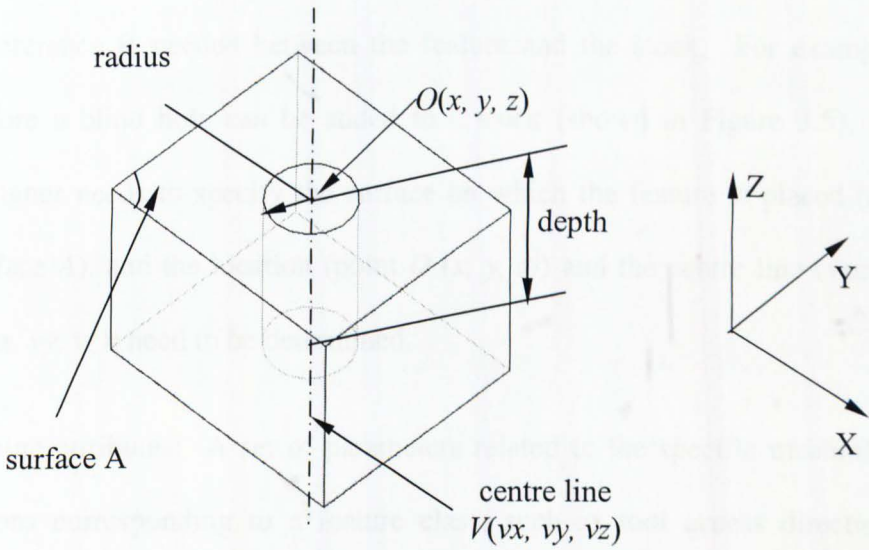


Figure 3.5 Geometric parameters of a hole

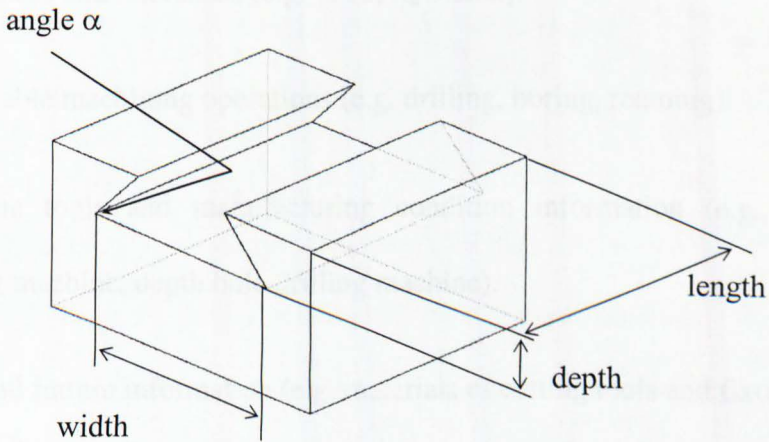


Figure 3.6 Geometric parameters of a slot

- Position and orientation parameters

Position and orientation parameters identify the spatial relationship between a feature and the stock. To establish the position and orientation, a reference is needed between the feature and the stock. For example, before a blind hole can be added to a stock (shown in Figure 3.5), the designer needs to specify the surface on which the feature is placed (i.e. surface A), and the location (point $O(x, y, z)$) and the centre line (vector $V(v_x, v_y, v_z)$) need to be determined.

3) Machining attributes: A set of parameters related to the specific machining operations corresponding to a feature class, such as tool access direction, machine tools, etc. As an example, considering process planning data for a through hole, it may include items such as the following:

- Roughness and tolerances (e.g. IT10, R_a 3.2 μ m).
- Applicable machining operations (e.g. drilling, boring, reaming).
- Machine tools and manufacturing condition information (e.g., radial drilling machine, depth hole drilling machine).
- Tool and fixture information (e.g. materials of cutting tools and fixtures).
- Tool access directions (e.g., direction I (0, 0, 1) and direction II (0, 0, -1)).
- Manufacturing cost information (e.g., operation cost, machine tool cost, tool and fixture cost, set-up change cost.)

4) **Validity constraints:** A set of constraints checked automatically at the feature instantiation stage to ensure the feature validity. There are three types of constraints: geometric and topological, machining, and interacting constraints. All these constraints are important for feature validation and their values can be decided using algebraic expressions (see pages 80-87), the manufacturing environment and features relationships.

- **Geometric and topological constraints**

Usually, these constraints appear as a standard range for specifying the size limits, which can be calculated using mathematical equations based on the shape parameters, class, and position and orientation of the feature. For example, the dimension of a hole cannot be larger than the size of the stock on which it is being placed; the depth of a blind hole must be restricted to be less than the size of the stock where the hole is to be added, otherwise the blind hole would become a through-hole.

- **Machining constraints**

It is possible that some features have valid geometric shapes and topology but still are invalid features because of their non-machinability. Different from other constraints, machining constraints mainly depend on the machining attributes of features and the specific workshop environment that features will be manufactured (e.g. machine tools can be available). For instance, long and thin holes may be regarded invalid if no machining methods are available for their manufacturing. At the design stage, the check for machining constraints is limited to constraints that can be

defined by algebraic expressions, e.g. the ratio of height to radius. Other machining attributes (e.g. tolerance and accuracy) are examined at the process planning stage, i.e. during selection of machining operations.

- Interacting constraints

Geometrical, topological and machining constraints are insufficient to fully retain feature validity when feature interactions occur. As known, feature interactions can cause serious constraint violations of valid feature instances. Therefore, the constraints for feature interactions must be defined, such as the dependent properties between parent and child features. An example is shown in Figure 3.7 that pocket B is added based on pocket A and becomes a child feature of pocket A. Due to this interacting constraint, pocket B will be invalid if pocket A is deleted. Unlike the other two constraints, there are no appropriate mathematical ways to determine the interacting validity. However, it is possible to develop some heuristic rules based on interacting rules for these constraints. These constraints are considered in a heuristic algorithm for interacting features recognition, which is discussed further in chapter 4.

5) UndiGraph: A face graph defining a feature pattern. It can be defined as

UndiGraph = (F , R), where

F is the finite non-NULL set of faces consisting of the feature:

$$F = \{face_i \mid face_i \in \text{Feature}\} \quad (3-4)$$

R is a set of relationships between faces: $R = \{FR\}$.

FR is a relationship with no specific direction between two faces:

$$FR = \{ \langle face_i, face_j \rangle \mid P(face_i, face_j) \wedge (face_i, face_j \in F) \} \quad (3-5)$$

$P(face_i, face_j)$ is a path with no specific direction between $face_i$ and $face_j$;

FR is symmetrical, i.e. $\langle face_x, face_y \rangle = \langle face_y, face_x \rangle$.

Figure 3.8 shows an example of UndiGraph.

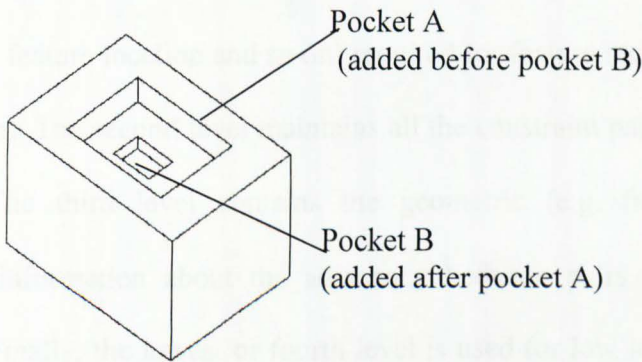


Figure 3.7 Example of interacting constraints

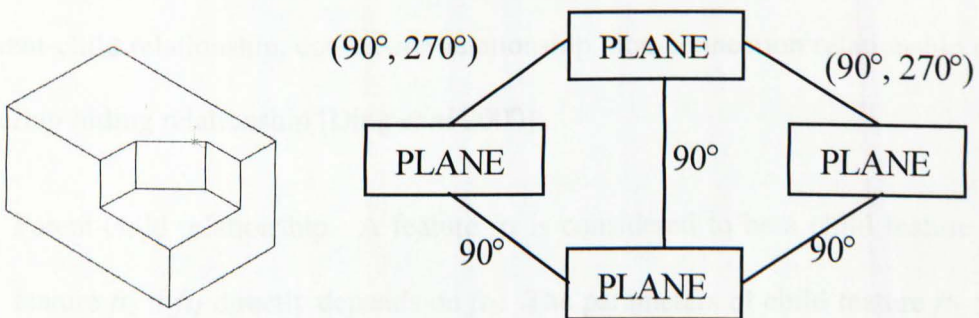


Figure 3.8 Example of feature Undigraph

3.4 Feature-based model

The proposed feature-based model allows the user to design a component directly with features predefined in the feature library. Unlike conventional geometrical models, it is designed for high-level data, i.e. feature instances. Therefore all the model operations are feature-based. As depicted in Figure 3.1, the highest level can be presented as a graph, where the nodes correspond to feature instances (e.g., $Feature_1$ and $Feature_n$) and the arcs store interaction relationships between feature pairs (e.g., 1 and 2). Figure 3.9 presents the detailed data structure of the proposed hierarchical feature-based model, whose data spans over four levels. The highest or the first level is about the basic entities and the information (such as feature relationship, feature location and so on) required for feature recognition and model manipulation. The second level maintains all the constraint parameters for feature validity. The third level contains the geometric (e.g. face attributes) and topological information about the adjacency between pairs of vertices, edges, faces, etc. Finally, the lowest or fourth level is used for low level geometric data such as tolerance, etc.

Four types of feature relationships are defined within the feature-based model: parent-child relationship, connection relationship, non-connection relationship and overlap-hiding relationship [Ding *et al* 2000].

- 1) Parent-child relationship. A feature ft_1 is considered to be a child feature of feature ft_2 if ft_1 directly depends on ft_2 . The parameters of child feature ft_1 are constrained by the parent feature, ft_2 . The parent feature, ft_2 is independent of feature ft_1 . In other words, the validity of the child feature, ft_1 needs to be re-

checked and necessary change made when the parent feature, ft_2 changes. On the contrary, the parent feature, ft_2 needs not change when the child feature, ft_1 changes.

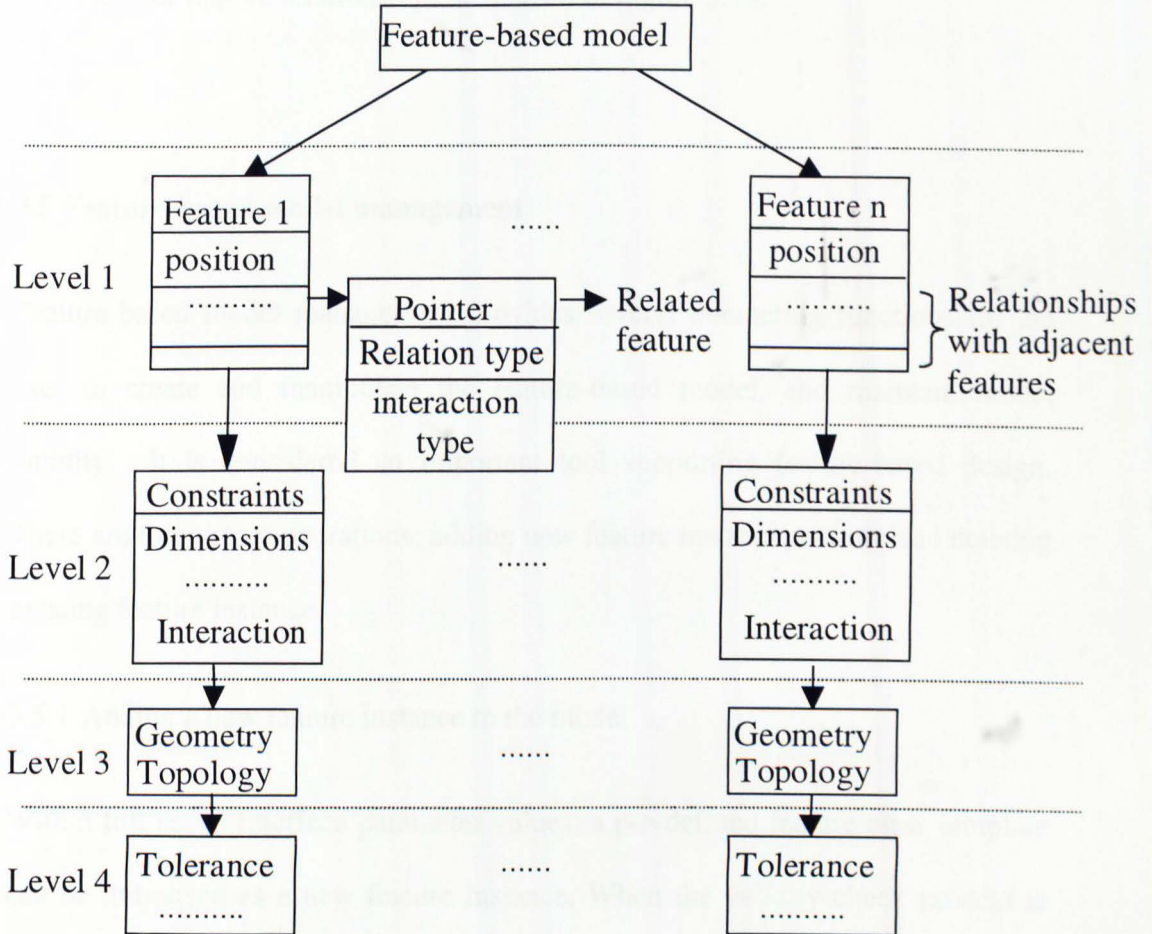


Figure 3.9 Structure of feature model

- 2) Connection relationship. If two features have a connection relationship, the validity of both features need not be re-checked. However, the relationship should be identified for downstream applications (e.g. CAPP).
- 3) Non-connection relationship. If there is a non-connection relationship between two features, they are not dependant on each other.

- 4) **Overlap-hiding relationship:** If a feature ft_1 is regarded as a hiding feature of feature ft_2 , it means ft_2 overlaps ft_1 completely. In this situation, ft_1 is deleted from the final result and only ft_2 is considered in the model.

Examples of feature relationships are shown in Figure 3.10.

3.5 Feature-based model management

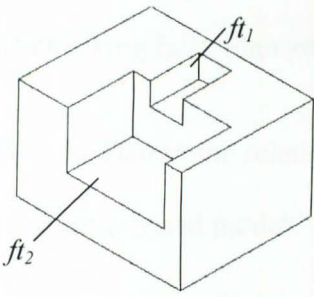
Feature-based model management provides several interacting functions for the user to create and manipulate the feature-based model, and maintain model validity. It is considered an important tool supporting feature-based design. There are three main operations: adding new feature instance, editing and deleting existing feature instance.

3.5.1 Adding a new feature instance to the model

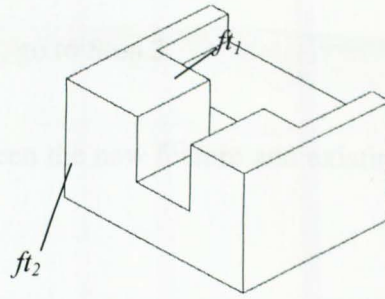
With a full set of interface parameter values, a pre-defined feature class template can be initialised as a new feature instance. When the validity check process is successfully performed, the new feature instance is added and the model data is updated. As shown in Figure D.1 (Appendix D), this process is detailed as:

Step 1: Input a set of parameters defining a feature.

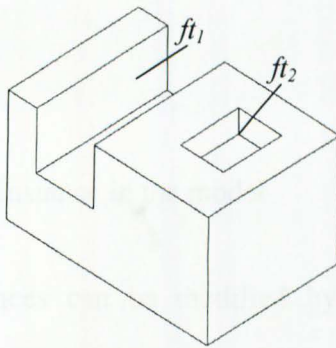
Step 2: Check the validity of the defined feature according to geometric and topological constraints using corresponding mathematical calculations described in section 3.5.4.



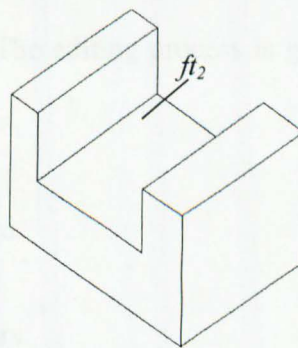
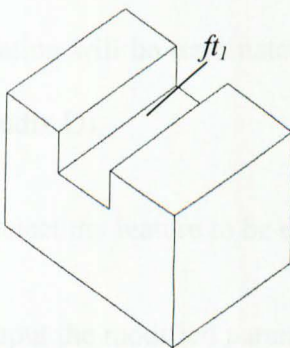
(a) Example of parent-child relationship
*ft*₁: blind slot (child)
*ft*₂: blind slot (parent)



(b) Example of connection relationship
*ft*₁: blind slot
*ft*₂: blind slot



(c) Example of non-connection relationship
*ft*₁: blind slot
*ft*₂: pocket



(d) Example of overlap-hiding relationship:
*ft*₁: through slot (hide)
*ft*₂: through slot

Figure 3.10 Examples of feature relationships

If the checking fails, then go to end. Else, go to Step 3.

Step 3: Determine relationships between the new feature and existing features in the feature-based model.

If any change (i.e. class or dimension change) is necessary, which user cannot agree, then go to end. Else, go to next step.

Step 4: Add the new feature instance and update the model.

Step 5: End.

3.5.2 Editing a feature instance in the model

Existing feature instances can be modified by changing the values of their parameters. The validity checking process consists of two parts: validity checking for the feature to be edited and validity checking for the existing features in the feature-based model. If the validity checking succeeds, the change to the feature is accepted and relevant parameters are updated accordingly; otherwise, the editing operation will be terminated. The editing process is given below (Figure D.2 in Appendix D).

Step 1: Select the feature to be edited.

Step 2: Input the modified parameters.

Step 3: Check the validity of the modified parameters using corresponding mathematical calculations.

If the checking fails, then go to end. Else, go to the next step.

Step 4: Re-check the features, which have a relationship with the feature to be edited. Two situations needs attention:

Situation 1: Features that had been changed class or dimension due to the addition of the feature to be edited. These features will be restored to their original classes or dimensions.

Situation 2: Features that are child features of the feature to be edited but their class or dimension had not been changed due to the addition of the feature to be edited. These features will be deleted, or they will be invalid.

If the user does not agree with above changes, then go to end. Else, go to step 5.

Step 5. Delete all previous relationships between the feature to be edited and existing features in the feature-based model.

Step 6. Determine new relationships between the feature to be edited and existing features in the feature-based model based on new editing parameters.

If the user doses not agree with any changes which are necessary, then go to step 7. Else, go to step 8.

Step 7. Restore all information of the feature-based model before this editing operation, including features, feature relationships, feature classes and feature dimensions. Then, go to end.

Step 8. Accept the modification and update the feature-based model.

Step 9. End.

Figure 3.7 gives an example where pocket A is the parent of pocket B. When the depth of pocket A is shortened, pocket B must be deleted. The request of editing will be cancelled if user does not agree with the deletion of pocket B.

3.5.3 Deleting a feature instance from the model

When a feature instance is deleted from the model, all its parameters and interaction relationships with other features will be deleted completely. This process is simpler than editing a feature as there is no need to identify new interacting relationships. However, all child features of the feature to be deleted must be checked:

- 1) Features that had been changed class or dimension due to the addition of the feature to be deleted. These features will be restored to their original classes or dimensions.
- 2) Features that are child features of the feature to be edited but their class or dimension had not been changed due to the addition of the feature to be deleted. These features will be deleted as well.

If the user does not agree with above changes, then the deleting operation will be cancelled. Figure D.3 in Appendix D describes the deleting process.

3.5.4 Checking validity of feature

As described in section 3.3, the restricting values of geometrical and topological constraints can be calculated by algebraic expressions on the feature parameters and the relevant dimensions of the stock. Tables 3.2 to 3.6 give some

mathematical expressions for constraints used in this research. The corresponding checking processes are implemented in the module using the C++ language.

Table 3.2 Calculating expressions for blind slot

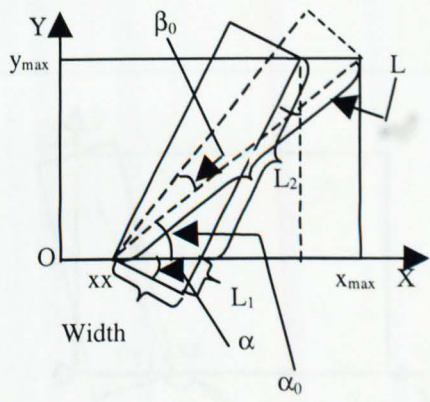
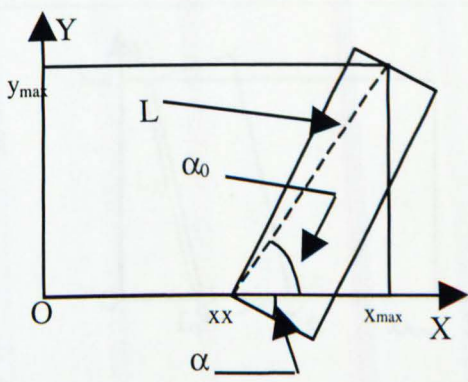
<p>Primitive variables</p>	$L = \sqrt{(x_{max} - xx)^2 + (y_{max} - oy)^2}$ $LA = \sqrt{(xx - o_x)^2 + (y_{max} - oy)^2}$ $\beta_0 = \arcsin(\text{width} / L) \quad (\alpha < 90^\circ)$ $\beta_0 = \arcsin(\text{width} / LA) \quad (\alpha > 90^\circ)$ $\alpha_0 = \text{arcctg} \left(\frac{ x_{max} - xx }{ y_{max} - oy } \right) \quad (\alpha < 90^\circ)$ $\alpha_0 = \text{arcctg} \left(\frac{ xx - o_x }{ y_{max} - oy } \right) \quad (\alpha > 90^\circ)$ <p>L_{max}: the maximum length W_{max}: the maximum width</p>
<p>Case 1</p>	<p>$(90^\circ - \alpha_0 - \beta_0) > \alpha \geq 0^\circ$</p> <p>$W_{max} = (x_{max} - xx) * \cos \alpha$</p> <p>$L_{max} = L_1 + L_2,$</p> <p>$L_1 = \text{width} * \text{tg} \alpha$</p> <p>$L_2 = (y_{max} - oy) / \cos \alpha$</p> 
<p>Case 2</p>	<p>$(90^\circ - \alpha_0) > \alpha \geq (90^\circ - \alpha_0 - \beta_0)$</p> <p>$W_{max} = (x_{max} - xx) * \cos \alpha$</p> <p>$L_{max} = L * \cos(90^\circ - \alpha - \alpha_0)$</p> 

Table 3.2 Calculating expressions for blind slot (continued)

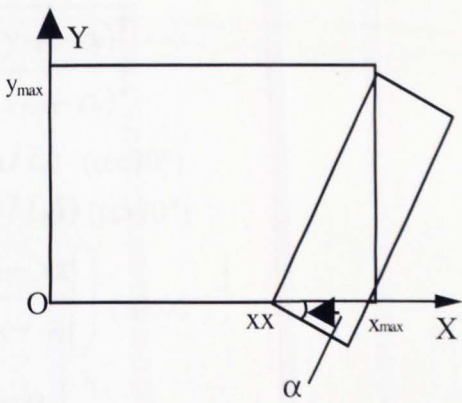
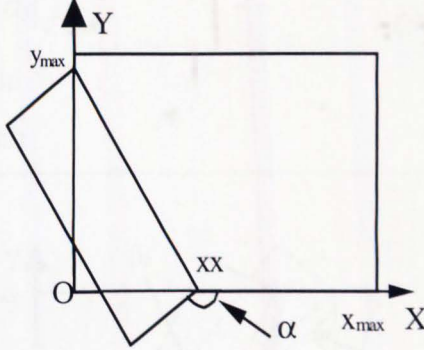
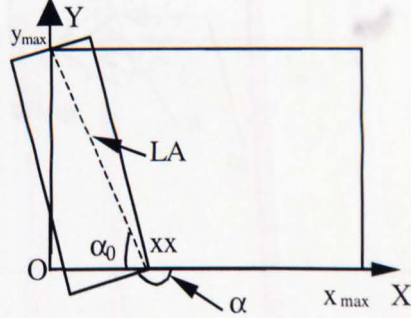
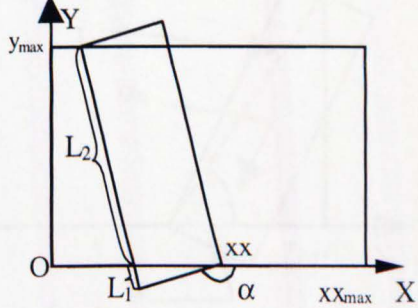
<p>Case 3</p>	 <p> $90^\circ > \alpha \geq (90^\circ - \alpha_0)$ $W_{\max} = (x_{\max} - xx) * \cos \alpha$ $L_1 = x_{\max} - xx$ $L_{\max} = L_1 / \cos(90^\circ - \alpha)$ </p>
<p>Case 4</p>	 <p> $(90^\circ + \alpha_0) > \alpha > 90^\circ$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_1 = xx - o_x$ $L_{\max} = L_1 / \cos(\alpha - 90^\circ)$ </p>
<p>Case 5</p>	 <p> $(90^\circ + \alpha_0 + \beta_0) > \alpha \geq (90^\circ + \alpha_0)$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_{\max} = LA * \cos(\alpha - 90^\circ - \alpha_0)$ </p>
<p>Case 6</p>	 <p> $180^\circ > \alpha \geq (90^\circ + \alpha_0 + \beta_0)$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_{\max} = L_1 + L_2,$ $L_1 = \text{width} * \text{tg}(180^\circ - \alpha)$ $L_2 = (y_{\max} - o_y) / \cos(180^\circ - \alpha)$ </p>

Table 3.3 Calculating expressions for through slot

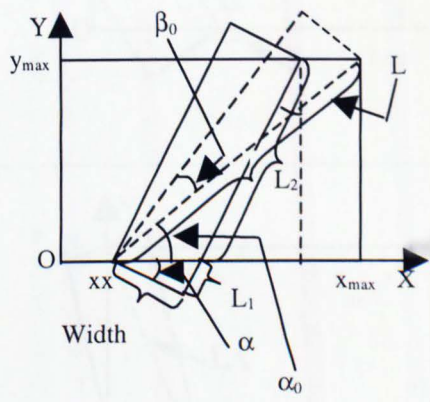
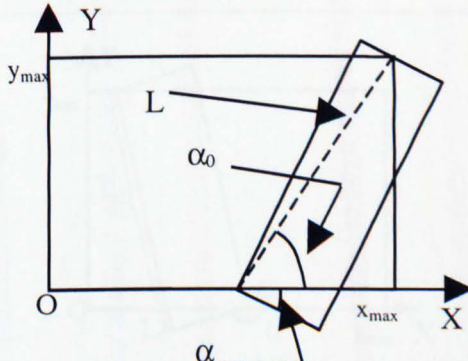
<p>Primitive variables</p>	$L = \sqrt{(x_{\max} - xx)^2 + (y_{\max} - oy)^2}$ $LA = \sqrt{(xx - o_x)^2 + (y_{\max} - oy)^2}$ $\beta_0 = \arcsin(\text{width} / L) \quad (\alpha < 90^\circ)$ $\beta_0 = \arcsin(\text{width} / LA) \quad (\alpha > 90^\circ)$ $\alpha_0 = \text{arcctg} \left(\frac{ x_{\max} - xx }{ y_{\max} - oy } \right) \quad (\alpha < 90^\circ)$ $\alpha_0 = \text{arcctg} \left(\frac{ xx - o_x }{ y_{\max} - oy } \right) \quad (\alpha > 90^\circ)$ <p>L_{\min}: the minimum length W_{\max}: the maximum width</p>
<p>Case 1</p>	<p>$(90^\circ - \alpha_0 - \beta_0) > \alpha \geq 0^\circ$</p> $W_{\max} = x_{\max} - xx \cdot \cos \alpha$ $L_{\min} = L_1 + L_2,$ $L_1 = \text{width} \cdot \text{tg} \alpha$ $L_2 = (y_{\max} - oy) / \cos \alpha$ 
<p>Case 2</p>	<p>$(90^\circ - \alpha_0) > \alpha \geq (90^\circ - \alpha_0 - \beta_0)$</p> $W_{\max} = x_{\max} - xx \cdot \cos \alpha$ $L_{\min} = L \cdot \cos(90^\circ - \alpha - \alpha_0)$ 

Table 3.3 Calculating expressions for through slot (continued)

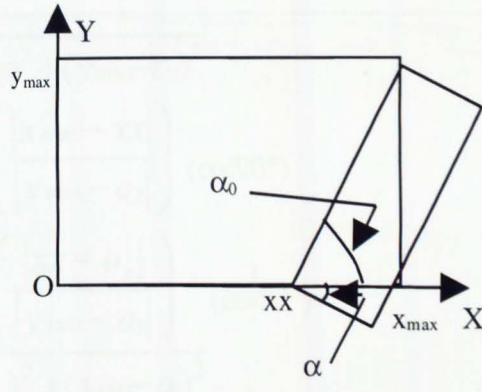
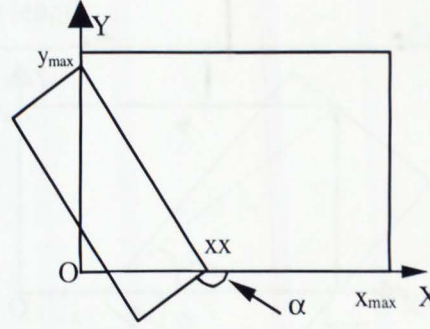
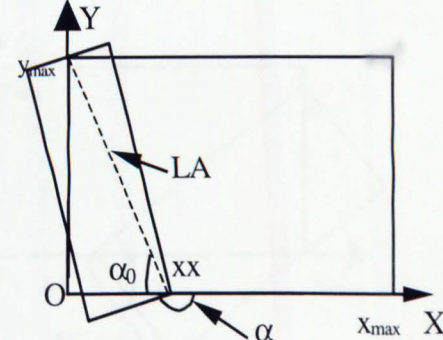
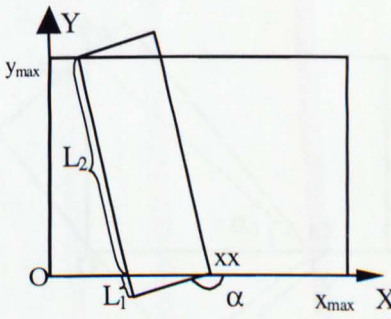
<p>Case 3</p>	<p> $90^\circ > \alpha \geq (90^\circ - \alpha_0)$ $W_{\max} = (x_{\max} - xx) * \cos \alpha$ $L_1 = x_{\max} - xx$ $L_{\max} = L_1 / \cos(90^\circ - \alpha)$ </p> 
<p>Case 4</p>	<p> $(90^\circ + \alpha_0) > \alpha > 90^\circ$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_1 = xx - o_x$ $L_{\min} = L_1 / \cos(\alpha - 90^\circ)$ </p> 
<p>Case 5</p>	<p> $(90^\circ + \alpha_0 + \beta_0) > \alpha \geq (90^\circ + \alpha_0)$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_{\min} = LA * \cos(\alpha - 90^\circ - \alpha_0)$ </p> 
<p>Case 6</p>	<p> $180^\circ > \alpha \geq (90^\circ + \alpha_0 + \beta_0)$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha)$ $L_{\min} = L_1 + L_2,$ $L_1 = \text{width} * \text{tg}(180^\circ - \alpha)$ $L_2 = (y_{\max} - o_y) / \cos(180^\circ - \alpha)$ </p> 

Table 3.4 Calculating expressions for through step

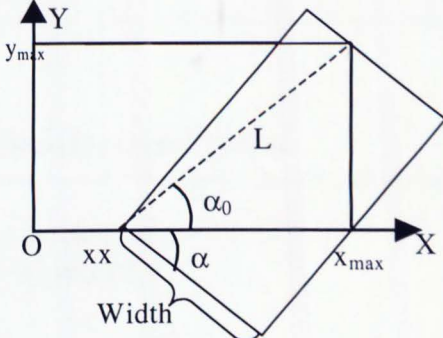
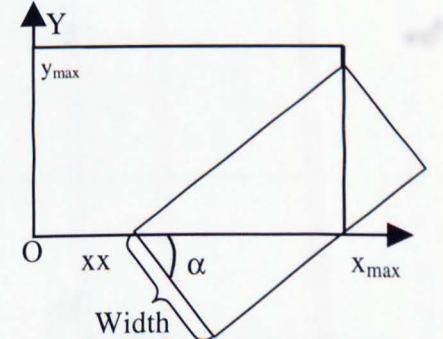
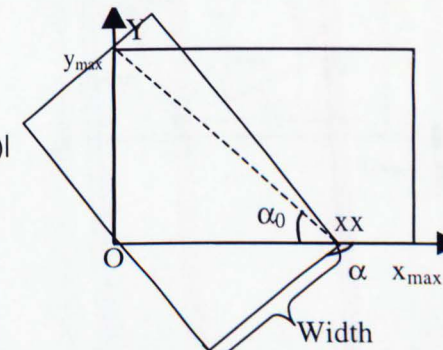
Primitive variables	$L = \sqrt{(x_{\max} - xx)^2 + (y_{\max} - o_y)^2}$ $\alpha_0 = \text{arcctg} \left(\frac{ x_{\max} - xx }{ y_{\max} - o_y } \right) \quad (\alpha < 90^\circ)$ $\alpha_0 = \text{arcctg} \left(\frac{ xx - o_x }{ y_{\max} - o_y } \right) \quad (\alpha > 90^\circ)$ $LA = \sqrt{(xx - o_x)^2 + (y_{\max} - o_y)^2}$ <p>L_{\min}: the minimum length</p>
Case 1	$(90^\circ - \alpha_0) > \alpha \geq 0^\circ$ $\text{Width} = (x_{\max} - xx) \cdot \cos \alpha $ $L_{\min} = L \cdot \cos(90^\circ - \alpha - \alpha_0)$ 
Case 2	$90^\circ > \alpha \geq (90^\circ - \alpha_0)$ $\text{Width} = (x_{\max} - xx) \cdot \cos \alpha $ $L_1 = x_{\max} - xx$ $L_{\min} = L_1 / \cos(90^\circ - \alpha)$ 
Case 3	$(90^\circ + \alpha_0) > \alpha > 90^\circ$ $\text{Width} = (xx - o_x) \cdot \cos(180^\circ - \alpha) $ $L_{\min} = LA \cdot \cos(\alpha - 90^\circ - \alpha_0)$ 

Table 3.4 Calculating expressions for through step (continued)

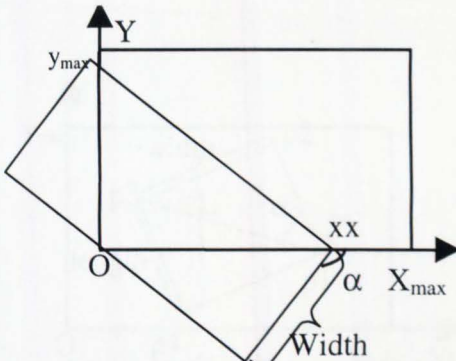
<p>Case 4</p>	$180^\circ > \alpha \geq (90^\circ + \alpha_0)$ $W_{\max} = (xx - o_x) * \cos(180^\circ - \alpha) $ $L_1 = xx - o_x$ $L_{\min} = L_1 / \cos(\alpha - 90^\circ)$ 
---------------	---

Table 3.5 Calculating expressions for closed pocket

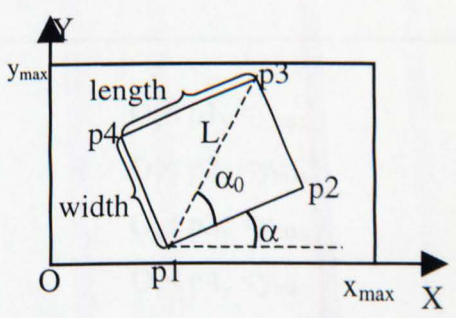
<p>Primitive variables</p>	$L = \sqrt{(\text{length})^2 + (\text{width})^2}$ $\alpha_0 = \text{arctg}\left(\frac{\text{width}}{\text{length}}\right)$
<p>Case 1</p>	$90^\circ > \alpha \geq 0^\circ$ $p2_x = p1_x + \text{length} * \cos\alpha$ $p2_y = p1_y + \text{length} * \sin\alpha$ $p3_x = p1_x + L * \cos(\alpha + \alpha_0)$ $p3_y = p1_y + L * \sin(\alpha + \alpha_0)$ $p4_x = p1_x - \text{width} * \cos(90^\circ - \alpha)$ $p4_y = p1_y + \text{width} * \sin(90^\circ - \alpha)$ 

Table 3.5 Calculating expressions for closed pocket (continued)

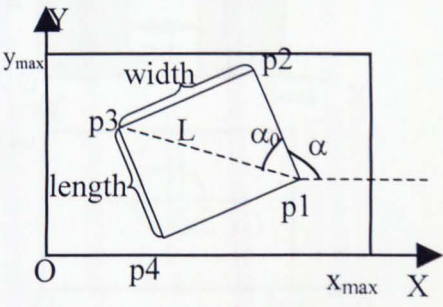
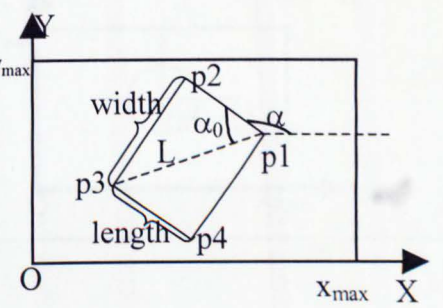
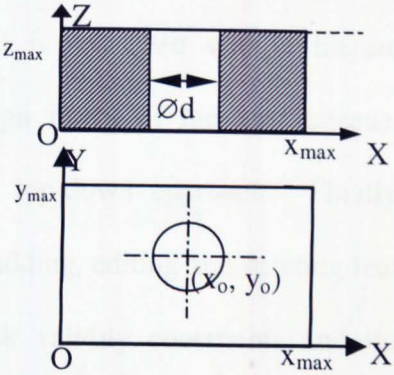
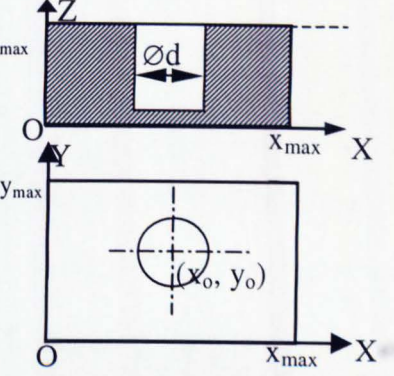
<p>Case 2</p>	<p>$(180^\circ - \alpha_0) > \alpha \geq 90^\circ$</p> <p>$p2_x = p1_x - \text{length} * \cos(180^\circ - \alpha)$ $p2_y = p1_y + \text{length} * \sin(180^\circ - \alpha)$ $p3_x = p1_x - L * \cos(180^\circ - \alpha_0 - \alpha)$ $p3_y = p1_y + L * \sin(180^\circ - \alpha_0 - \alpha)$ $p4_x = p1_x - \text{width} * \cos(\alpha - 90^\circ)$ $p4_y = p1_y - \text{width} * \sin(\alpha - 90^\circ)$</p>	
<p>Case 3</p>	<p>$180^\circ > \alpha \geq (180^\circ - \alpha_0)$</p> <p>$p2_x = p1_x - \text{length} * \cos(180^\circ - \alpha)$ $p2_y = p1_y + \text{length} * \sin(180^\circ - \alpha)$ $p3_x = p1_x - L * \cos(\alpha + \alpha_0 - 180^\circ)$ $p3_y = p1_y - L * \sin(\alpha + \alpha_0 - 180^\circ)$ $p4_x = p1_x - \text{width} * \cos(\alpha - 90^\circ)$ $p4_y = p1_y - \text{width} * \sin(\alpha - 90^\circ)$</p>	
<p>Value limitations</p>	<p>$O_x < p1_x < x_{\max}$ $O_x < p2_x < x_{\max}$ $O_x < p3_x < x_{\max}$ $O_x < p4_x < x_{\max}$</p>	<p>$O_y < p1_y < y_{\max}$ $O_y < p2_y < y_{\max}$ $O_y < p3_y < y_{\max}$ $O_y < p4_y < y_{\max}$</p>

Table 3.6 Calculating expressions for blind/through hole

through hole	$\text{depth} \geq z_{\max} - O_z $ $x_0 + d/2 < x_{\max}$ $x_0 - d/2 > O_x$ $y_0 + d/2 < y_{\max}$ $y_0 - d/2 > O_y$ $\text{depth}/d < R_{\text{limit}}$ R_{limit} is determined by manufacturing environment	
blind hole	$\text{depth} < z_{\max} - O_z $ $x_0 + d/2 < x_{\max}$ $x_0 - d/2 > O_x$ $y_0 + d/2 < y_{\max}$ $y_0 - d/2 > O_y$ $\text{depth}/d < R_{\text{limit}}$	

3.6 Summary

This chapter has described the design by features module. Firstly, a new architecture has been introduced for design by features with a feature library, feature-based model, feature library management and feature-based model management. Secondly, a classification scheme has been presented for manufacturing features based on the ISO STEP standard. The main characteristic of the classification is the consideration of STEP AP224 and a viewpoint of both manufacturing and design. Thirdly, a data structure for the feature class has been

designed. Although at the current stage, the work deals only with primitive features, new feature classes can be defined according to the defined data structure. Fourthly, a feature-based model is described with a hierarchical structure. The model can reflect the design intent of the user because the modelling history can be traced through a top-down approach. Finally, the feature-based model management deals with adding, editing and deleting features. Mathematical calculations are used to check validity constraints and thus to effectively maintain the feature validity in terms of geometry and topology.

Chapter 4.

Interacting Features Recogniser

Although design by features does not need to recognise the primitive features defined in the feature library, it is imperative to apply feature recognition techniques to deal with interacting features. Figures 4.1 and 4.2 show two situations of interacting features, which lead to different results. Most methods for interacting feature recognition used to date are based on analysing the geometrical and topological information of the new volume created by all of the interacting features, such as the Graph Matching showing in Figures 4.1 and 4.2. They are possibly successful in some conditions, e.g. two simple interacting features which may be merged to one feature. However, these methods are not very efficient or useful in many situations. For instance, the blind slot and the closed pocket shown in Figure 4.3 should not be recognised as a whole entity though they interact. This research presents a new identification algorithm of interacting features aiming to overcome the drawbacks of existing methods.

Comparing the two illustrations shown in Figures 4.3 and 4.4, it can be seen that the two situations have different interacting entities, *A* and *B*. In Figure 4.4, the interacting entity *B* has a face *f1'* which represents the entire face *f1* of feature *B2* (closed pocket), which causes the loss of a face of feature *B2* completely and consequently a class change to feature *B2*. On the contrary, in Figure 4.3, the interacting entity *A* does not contain a face similar to face *f1'* in the interacting entity *B*. Therefore, the validity of feature *A1* and *A2* remains and no further

recognition is needed. In other words, it is possible to solve the interacting problem by analysing the interacting entity because different interacting entities will lead to different interacting results. Thus, unlike conventional approaches directly recognising the new volume created by all interacting features, the proposed heuristic algorithm focuses on analysing the relationships between pairs of features and determines an appropriate manipulation (e.g. merge, divide, class change). This avoids unnecessary feature recognition work and resolves interacting features more efficiently.

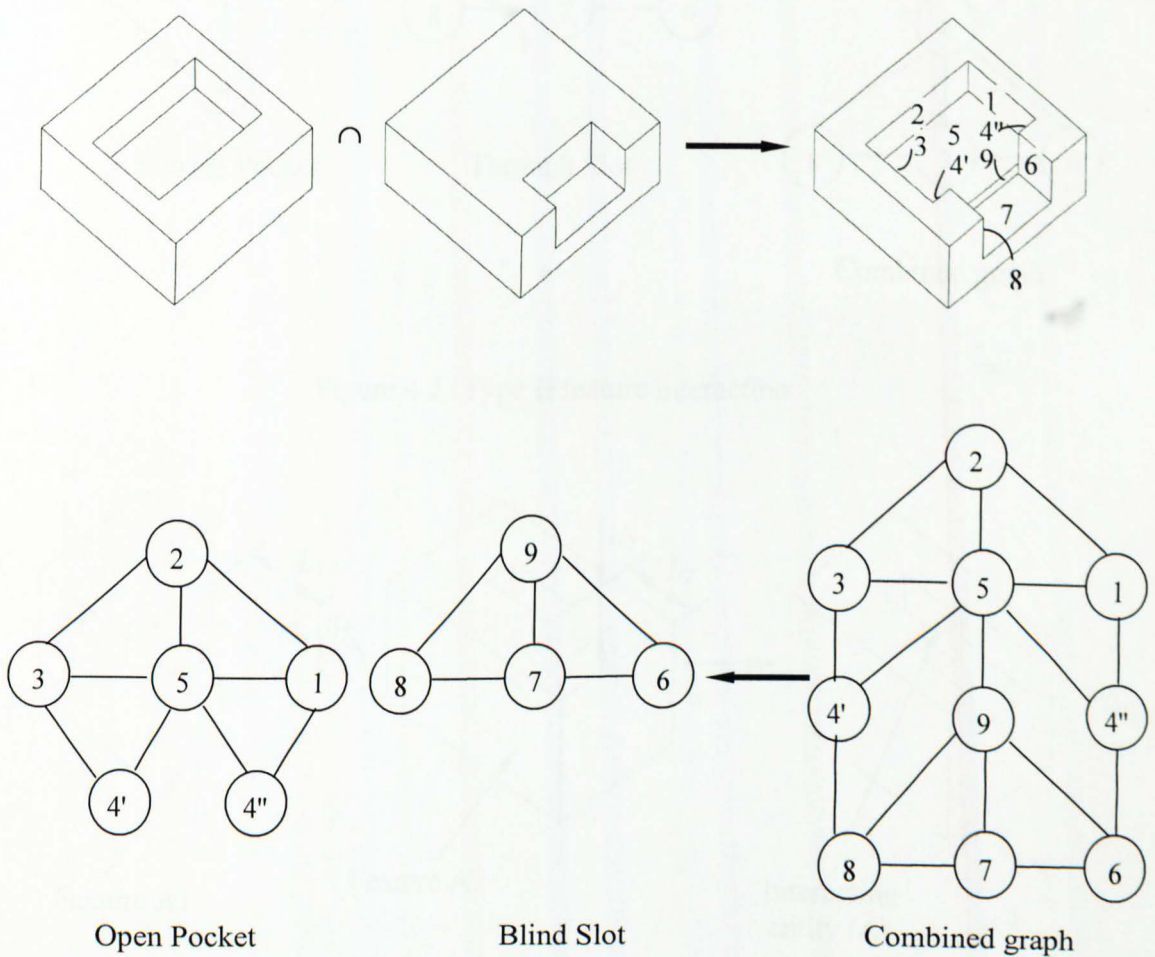


Figure 4.1 Type I feature interaction

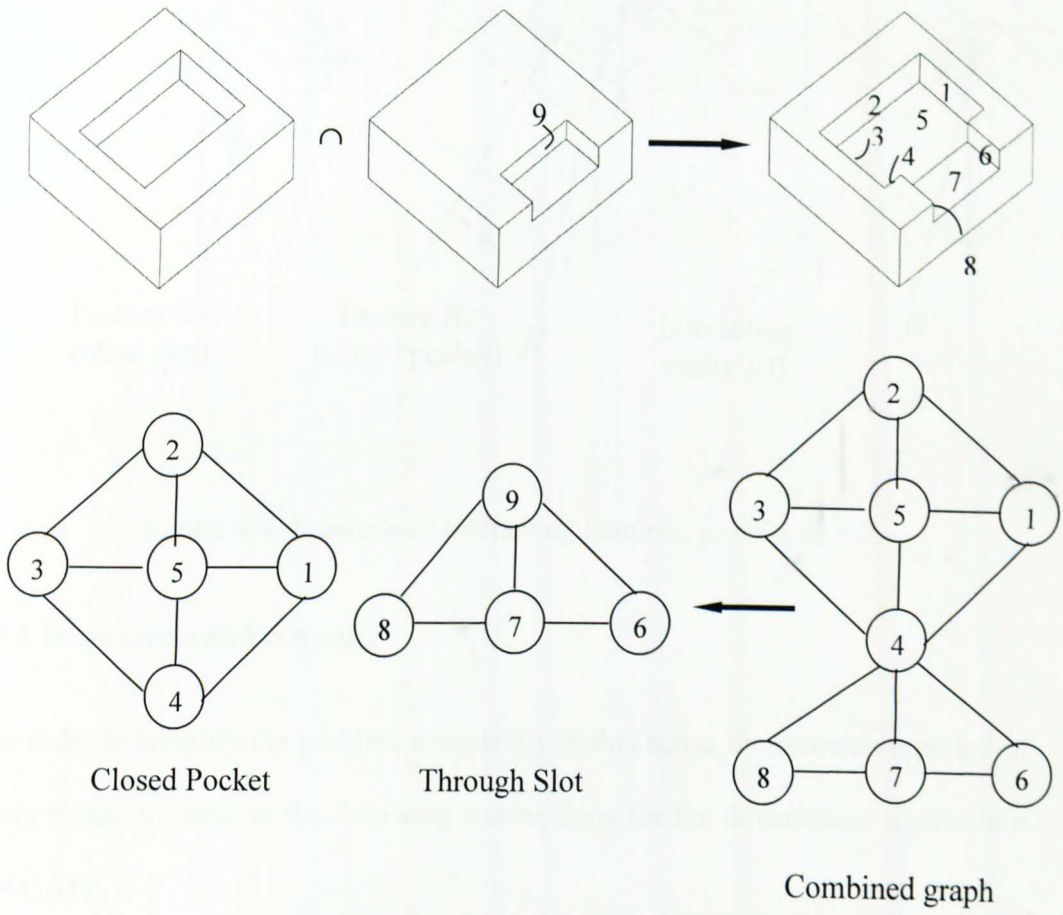


Figure 4.2 Type II feature interaction

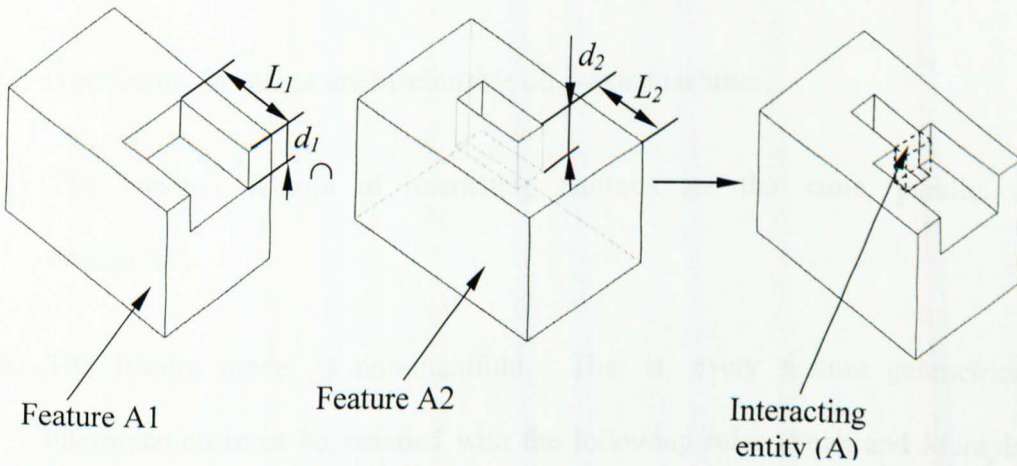


Figure 4.3 Example of interacting features, $L_1 > L_2, d_1 < d_2$

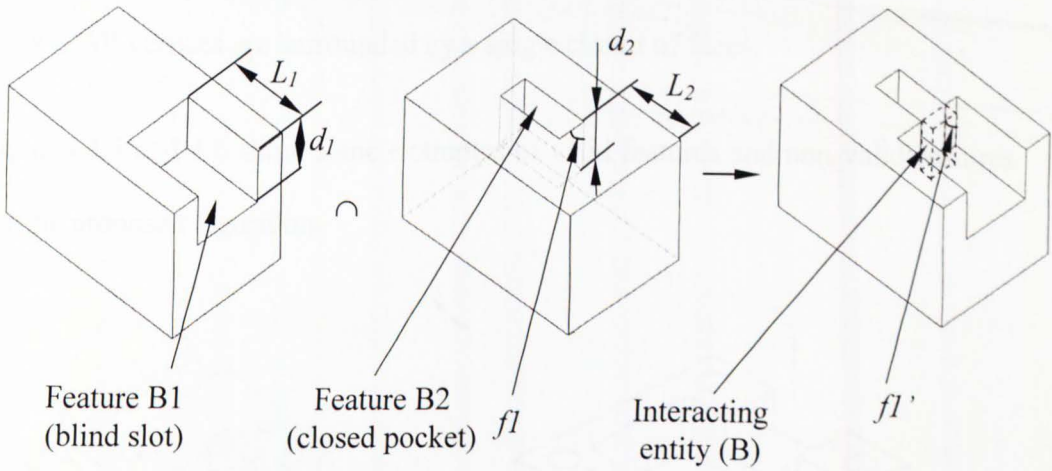


Figure 4.4 Example of interacting features, $L_1 > L_2$, $d_1 = d_2$

4.1 Basic terms and concepts

In order to simplify the problem complexity at this stage, the research reported in this thesis is based on the following assumptions for the downstream application of CAPP.

- 1) The features belong to the internal features defined in chapter 3, such as slot, pocket, and so on.
- 2) The feature instances are machinable on 3-axis machines.
- 3) The location vectors of interacting features are the same, parallel or orthogonal.
- 4) The feature model is non-manifold. That is, every feature geometrical interpretation must be satisfied with the following rules [Shah and Mäntylä, 1995]:
 - All edges separate exactly two faces

- All vertices are surrounded by a single circuit of faces.

Figures 4.5 and 4.6 show some examples of valid features and non-valid features for the proposed algorithm.

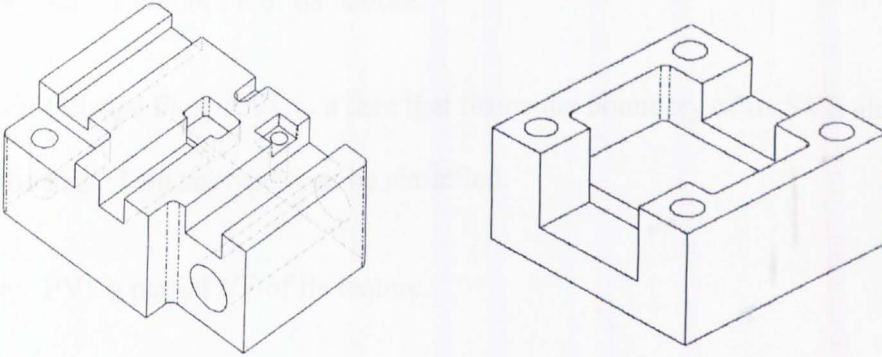


Figure 4.5 Valid components for this research

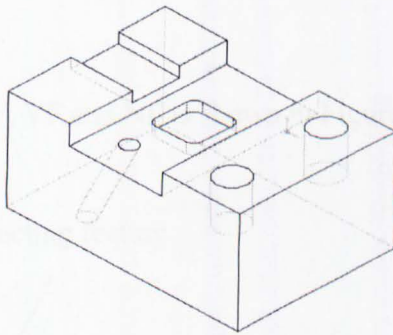


Figure 4.6 Invalid component due to a sloping hole

A number of terms are used for describing the interacting features recognition algorithm.

- 1) *SVE* (Spatial Virtual Entity): *SVE* is defined as an entity, which is equivalent to the volume removed from the initial material stock to obtain the final boundary of a feature.

2) *FF* (Feature Face): *FF* is a face that physically constitutes the basic shape of a feature on the model. It can be further classified into two types.

- *PF*: a partial *FF* of its feature.
- *CF*: a whole *FF* of its feature.

3) *VF* (Virtual Face): *VF* is a face that forms the boundary of its *SVE* along with the *FF*s. Two subtypes can be classified.

- *PV*: a partial *VF* of its feature.
- *CV*: a whole *VF* of its feature.

4) *NF* (None Face): *NF* is an internal face in *SVE* and does not constitute the boundary of *SVE*.

An example of *SVE*, *FF*, *VF* and *NF* is shown in Figure 4.7.

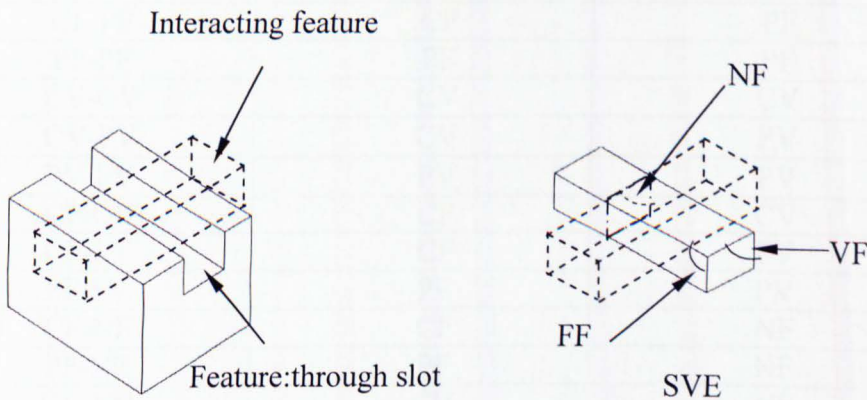
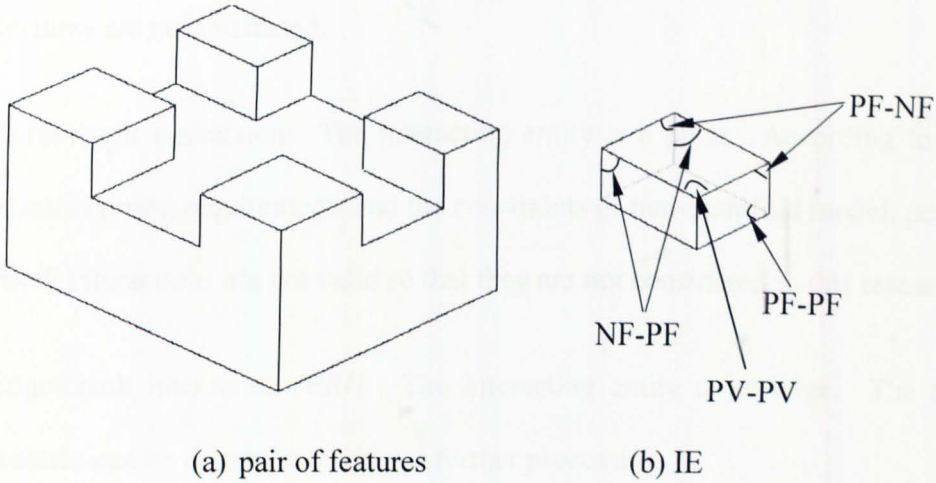


Figure 4.7 Example of *SVE*, *FF*, *VF* and *NF*

5) *IE* (Interacting Entity): *IE* is an entity defining the intersection of a features pair. Figure 4.8 shows an example of *IE*. Based on the above definitions, there are thirteen types of faces for *IE* (shown in Table 4.1).

Figure 4.8 Example of *IE*Table 4.1. Face types for *IE*

The type of face in <i>IE</i>	The type of face of Feature ft_1	The type of face of Feature ft_2
CF-CF	CF	CF
CF-PF	CF	PF
PF-PF	PF	PF
CV-CV	CV	CV
CV-PV	CV	PV
PV-PV	PV	PV
CF-CV	CF	CV
CF-PV	CF	PV
PF-PV	PF	PV
CF-NF	CF	NF
PF-NF	PF	NF
CV-NF	CV	NF
PV-NF	PV	NF

4.2 The types of feature interactions

The following five types of interactions are normally encountered:

- 1) Non-result interaction: The interacting entity is none, which means the two features are not interacted.
- 2) Point-result interaction: The interacting entity is a point. According to the manufacturing requirements and the constraints of non-manifold model, point-result interactions are not valid so that they are not considered in this research.
- 3) Edge-result interaction (*ERI*): The interacting entity is an edge. The two features can be determined without further processing.
- 4) Face-result interaction (*FRI*): The interacting entity is one face. It can be further classified into four types: CF-CF, CF-PF (or PF-CF), PF-PF and PF-CV (or CV-PF).
 - CF-CF: For either feature ft_1 or feature ft_2 , the types of the interacting face are both CF faces.
 - CF-PF (or PF-CF): The intersecting face appears as a CF face of feature ft_1 while a PF face of feature ft_2 .
 - PF-PF: The intersecting face is regarded as a PF face for both features.
 - PF-CV (or CV-PF): The intersecting face represents a PF face of feature ft_1 and a CV face of feature ft_2 .

- 5) Volume-result interaction (*VRI*): The interacting entity is a volume. This type of intersection is more complicated and needs to be analysed further.

Examples of the above interacting situations are shown in Figures 4.9 to 4.20.

4.3 Herustics algorithm for interacting feature recognition

The basic mechanism of the proposed algorithm is to make use of *IEs* of pairs of interacting features. The first step is to traverse all features in the hierarchical feature-based model. Then, Boolean set intersections are performed on each feature pair. Finally, the *IE* is analysed till a relationship is determined for each pair of features. The algorithm is described below.

Assume two features, *A* and *B* satisfy the conditions mentioned before, whose interacting entity is $IE_{AB} = SVE_A \cap SVE_B$. Four possible situations exist.

Situation 1: The interacting entity is none: $IE_{AB} = \emptyset$ (e.g. Figure 4.9).

The relationship between feature *A* and feature *B* has a non-connection relationship. No further recognition is needed.

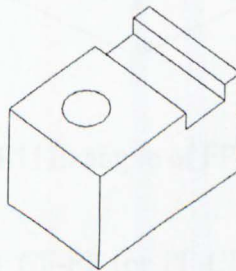


Figure 4.9 Example of $IE_{AB} = \emptyset$

Situation 2: The interacting entity is an edge: $IE_{AB} = \text{edge}$.

The two features have a connection relationship (e.g. Figure 4.10). Although the two features share one edge, their individual validity is not affected. Therefore, there is no further feature re-recognition.

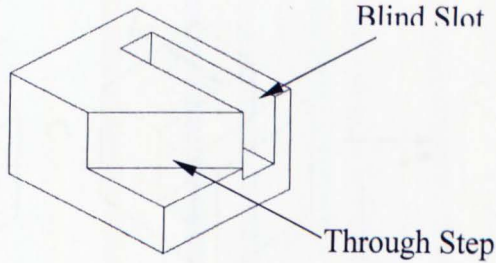


Figure 4.10 Example of ERI

Situation 3: The interacting entity is a face: $IE_{AB} = \text{face}$.

Step 1: If $IE_{AB} = \text{face} \in \text{PF-PF}$, the relationship between the two features is a connection relationship (e.g. Figure 4.11). The validity of both features is not affected though there may be some impact on their machining planning.

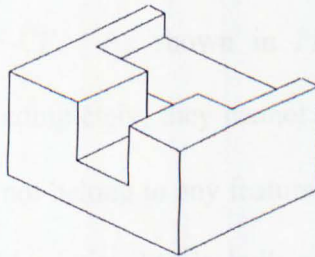


Figure 4.11 Example of FRI (PF-PF)

Step 2: If $IE_{AB} = \text{face} \in \text{CF-PF}$ (or PF-CF), a parent-child relationship is built with a feature class changed. For example, in Figure 4.12, a blind slot (A)

intersects with a through slot (B). The interacting entity $IE_{AB} = CF_{\text{blind-slot}}$ and $IE_{AB} = PF_{\text{through-slot}}$, which means a FF face in the blind slot (A) disappears completely due to the interaction. Thus, the class of feature A (blind slot) changes to a through slot which is a child feature of feature B (through slot).

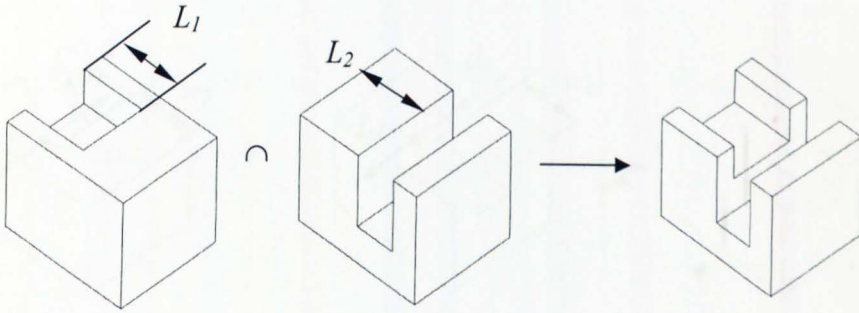


Figure 4.12 Example of $IE_C = \text{face} \in CF\text{-}PF$, $L_1 = L_2$

Step 3: If $IE_{AB} = \text{face} \in CF\text{-}CF$, the two features are merged into a new feature class or a parent-child relationship is identified according to the requirements of merger. For instance, two interacting blind slots shown in Figure 4.13(a) can be merged into one new through slot and become hiding features of the new feature (a through slot). Another example is two pockets intersecting a $\text{face} \in CF\text{-}CF$. As shown in Figure 4.13(b), although both features lose a FF face completely, they cannot be unite together because the new shape created does not belong to any feature class defined in this research. Therefore, a parent-child relationship is built and one of them becomes the child feature with a new feature class – through slot. A neural network-based feature recogniser has been designed to analyse the possibility of merger, which is described in chapter 5.

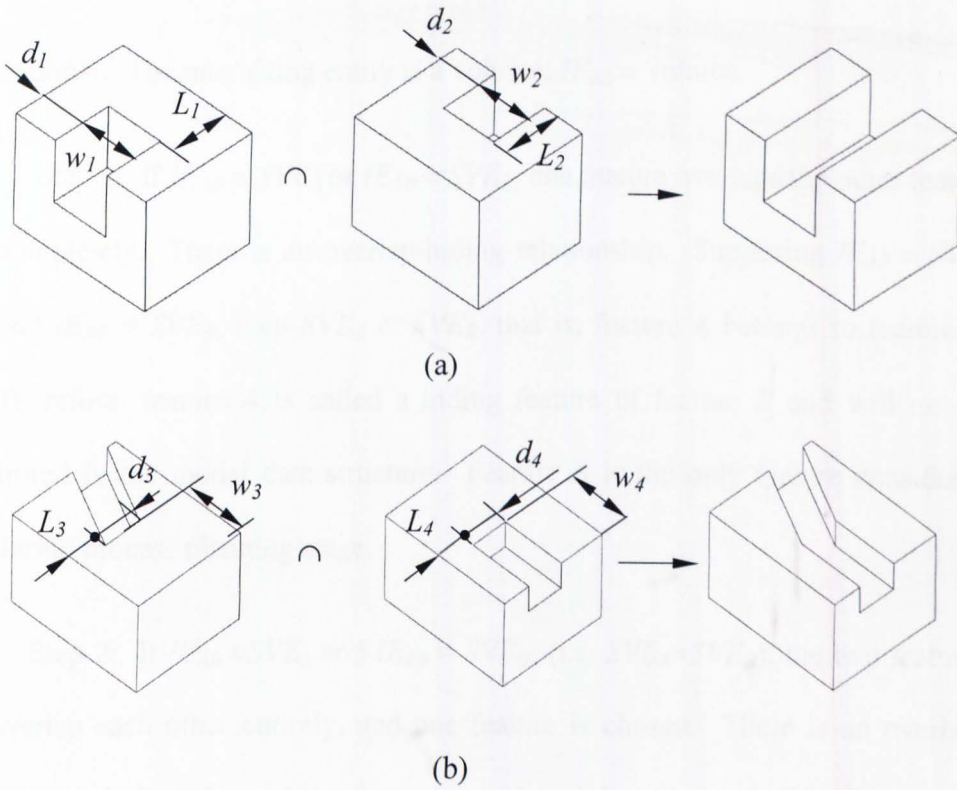


Figure 4.13 Examples of $IE_C = \text{face} \in \text{CF-CF}$, $IE_C = \text{CF}_A$, $IE_C = \text{CF}_B$

(a) $L_1 = L_2, d_1 = d_2, w_1 = w_2$

(b) $L_3 = L_4, d_3 = d_4, w_3 = w_4$

Step 4: If $IE_{AB} = \text{face} \in \text{CV-PF}$ (or PF-CV), there is a parent-child relationship between the two features. A good example of two interacting pockets is shown in Figure 4.14, which $IE_{AB} = \text{CV}_I$ and $IE_{AB} = \text{PF}_{II}$.

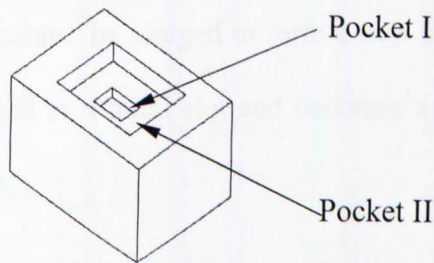


Figure 4.14 Examples of $IE_C = \text{face} \in \text{CV-PF}$, $IE_C = \text{CV}_I$, $IE_C = \text{PF}_{II}$

Situation 4: The interacting entity is a volume: $IE_{AB} = \text{volume}$.

Step 1: If $IE_{AB} = SVE_A$ or $IE_{AB} = SVE_B$, one feature overlaps the other feature completely. There is an overlap-hiding relationship. Supposing $IE_{AB} = SVE_A$ and $IE_{AB} \neq SVE_B$, then $SVE_A \subset SVE_B$, that is, feature A belongs to feature B . Therefore, feature A is called a hiding feature of feature B and will not be stored in the model data structure. Feature B is the only feature considered during process planning stage.

Step 2: If $IE_{AB} = SVE_A$ and $IE_{AB} = SVE_B$, (i.e. $SVE_A = SVE_B$), the two features overlap each other entirely, and one feature is chosen. There is an overlap-hiding relationship, and the feature not chosen becomes a hiding feature and does not being considered later.

Step 3: If $f_i \in IE_{AB}$, $f_i \in \text{CF-NF}$ (or NF-CF), $f_j \in IE_{AB}$ and $f_j \in \text{PF-PF}$, they are merged to form a new feature if they meet the requirements of merger, i.e. the new feature belongs to a certain feature class that the system can recognise. Otherwise, a parent-child relationship is detected. Figure 4.15 shows an example. The blind hole and the blind slot are merged into a new feature: blind slot (`radiused_slot_end_type`). Another instance is shown in Figure 4.16 where feature A and feature B cannot be merged to form a new feature, the feature B (closed pocket) is changed to a blind slot and becomes a child feature of the feature A (parent feature).

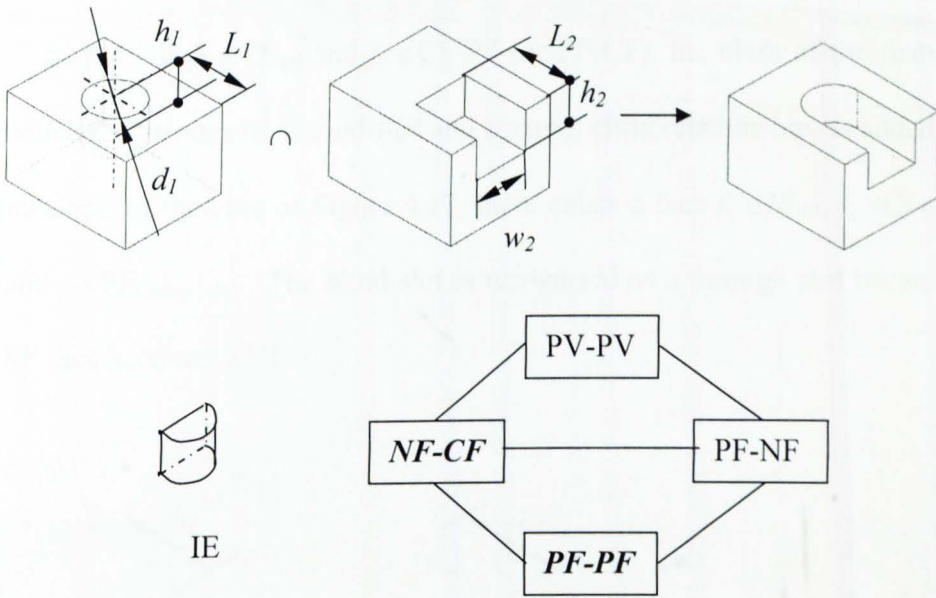


Figure 4.15 Example1 of $IE_{AB} = \text{Volume}$,
 $f_i \in IE_{AB}$, $f_j \in IE_{AB}$, $f_i \in CF-NF$, $f_j \in PF-PF$,
 $L_1 = L_2$, $d_1 = w_2$, $h_1 = h_2$

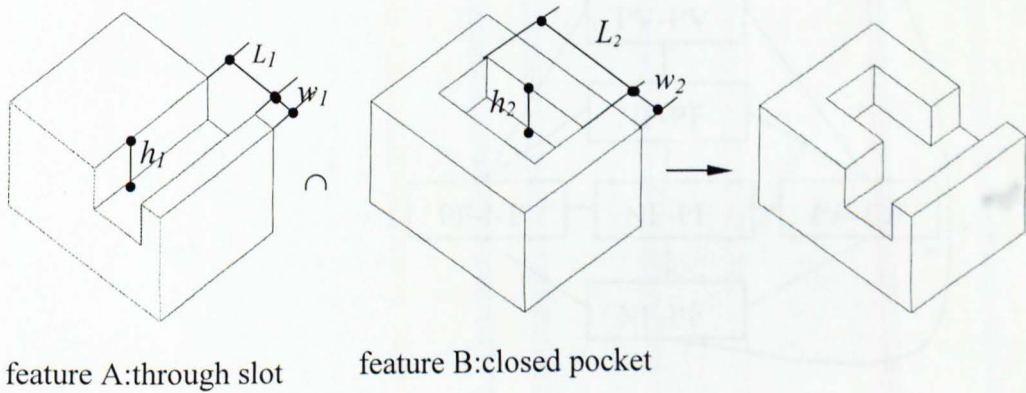


Figure 4.16 Example2 of $IE_{AB} = \text{Volume}$,
 $f_i \in IE_{AB}$, $f_j \in IE_{AB}$, $f_i \in NF-CF$, $f_j \in PF-PF$
 $L_1 + w_1 < L_2 + w_2$, $w_2 > w_1$, $h_1 = h_2$

Step 4: If $f_i \in IE_{AB}$ and $f_i \in CF\text{-}PF$ (or $PF\text{-}CF$), the class of the feature to which CF belongs to, is modified and a parent-child relationships is added. For instance, in the case of Figure 4.17, there exists a face $f_i \in IE_{AB}$, $f_i = CF_{\text{blind-slot}}$ and $f_i = PF_{\text{through-slot}}$. The blind slot is recognised as a through slot because the FF face becomes a VF .

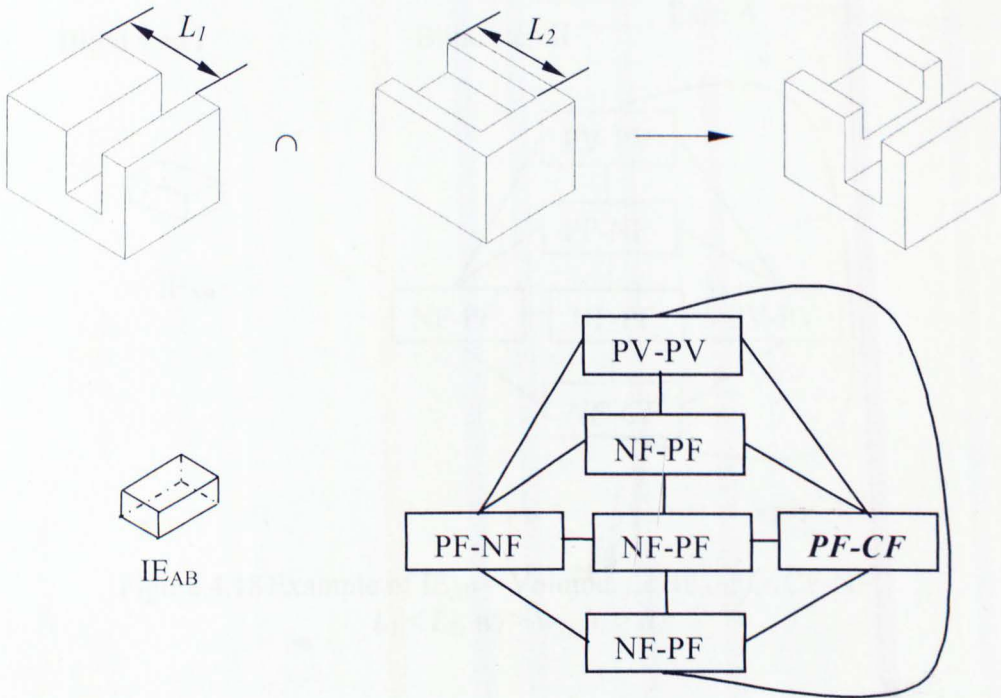


Figure 4.17 Example of $IE_{AB} = \text{Volume}$,
 $f_i \in IE_{AB}$, $f_i \in PF\text{-}CF$, $L_1 = L_2$

Step 5: If $f_i \in IE_{AB}$, $f_i \in CF\text{-}NF$ (or $NF\text{-}CF$), $f_j \in IE_{AB}$, and $f_j \notin PF\text{-}PF$, e.g. $f_i = CF_A$ and $f_j = NF_B$, then the class of feature A is changed and the two features have a parent-child relationship, i.e. feature A becomes a child feature of feature B. For example, two blind steps (I and II) intersecting in Figure 4.18, their interacting entity IE_{AB} contains a $NF\text{-}CF$ face and does not include a $PF\text{-}$

PF face; the class of blind step II changes through step due to loss of an entire face A (CF). The through step is a child feature of blind step I.

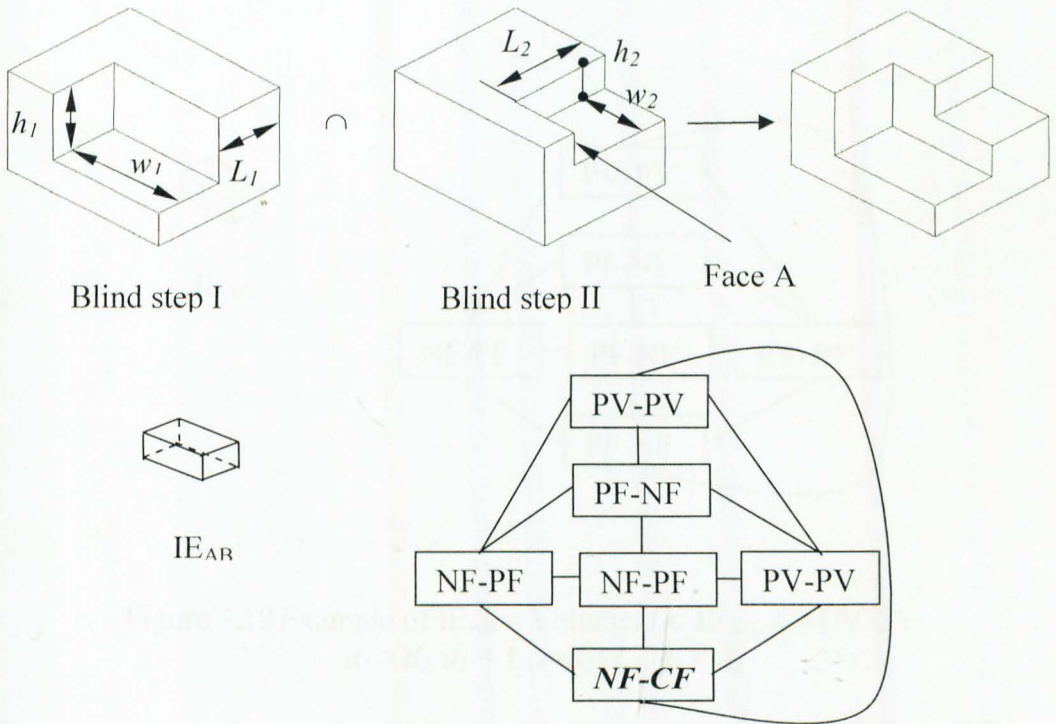


Figure 4.18 Example of $IE_{AB} = \text{Volume}$, $f_i \in IE_{AB}$, $f_i \in CF-NF$
 $L_1 < L_2$, $w_1 > w_2$, $h_1 > h_2$

Step 6: If $f_i \in IE_{AB}$ and $f_i \in CV-PV$ (or $PV-CV$), then the dimension of the feature with CV is modified while the feature class remains unchanged. Based on this modification, a parent-child relationship is built. An example is a through slot interacting with a blind slot (Figure 4.19), where $f_i \in IE_{AB}$, $f_i = CV_{\text{through-slot}}$ and $f_i = PV_{\text{blind-slot}}$. In this case, the through slot is shortened and becomes a child feature of the blind slot.

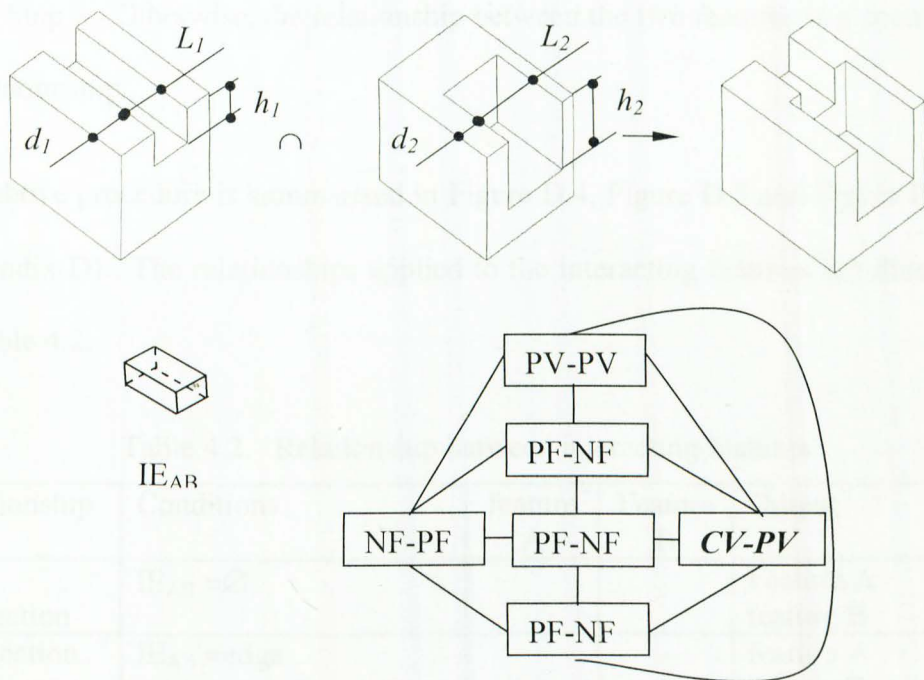


Figure 4.19 Example of $IE_{AB} = \text{Volume}$, $f_i \in IE_{AB}$, $f_i \in CV-PV$
 $d_1 > d_2$, $d_1 + L_1 < d_2 + L_2$, $h_1 < h_2$

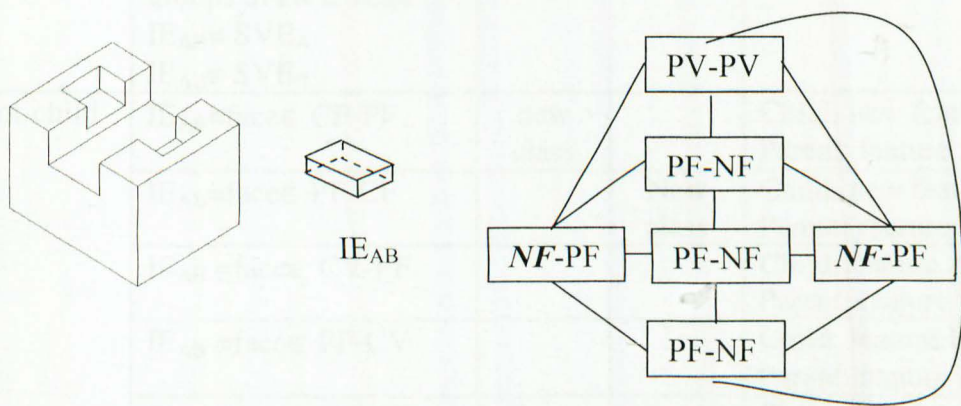


Figure 4.20 Example of $IE_{AB} = \text{Volume}$ with two non-connected NFs

Step 7: If IE_{AB} has two non-connected groups of NF in one feature (say feature A), feature A is divided into two parts. Figure 4.20 shows a pocket (with two non-connected NFs) divided into two blind slots by a through slot.

Step 8: Otherwise, the relationship between the two features is a connection relationship.

The above procedure is summarised in Figure D.4, Figure D.5 and Figure D.6 (in Appendix D). The relationships applied to the interacting features are illustrated in Table 4.2.

Table 4.2. Relationship between interacting features

Relationship	Conditions	feature A	Feature B	Output
Non-connection	$IE_{AB} = \emptyset$			Feature A feature B
Connection	$IE_{AB} = \text{edge}$			feature A feature B
	$IE_{AB} = \text{face} \in \text{PF-PF}$			feature A feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \notin \text{CF-X}, f_i \notin \text{CV-X},$ two non-connected groups of $NF \notin IE_{AB}$ $IE_{AB} \neq SVE_A$ $IE_{AB} \neq SVE_B$			feature A feature B
Parent-child	$IE_{AB} = \text{face} \in \text{CF-PF}$	new class		Child: new feature A' Parent: feature B
	$IE_{AB} = \text{face} \in \text{PF-CF}$		New class	Child: new feature B' Parent: feature A
	$IE_{AB} = \text{face} \in \text{CV-PF}$			Child: feature A Parent: feature B
	$IE_{AB} = \text{face} \in \text{PF-CV}$			Child: feature B Parent: feature A
	$IE_{AB} = \text{face} \in \text{CF-CF}$ Out of requirements of merger	new class		Child: new feature A' Parent: feature B
	$IE_{AB} = \text{face} \in \text{CF-CF}$ Out of requirements of merger		new class	Child: new feature B' Parent: feature A
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{CF-PF}$	new class		Child: new feature A' Parent: feature B

	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{PF-CF}$		new class	Child: new feature B' Parent: feature A
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{CF-NF}$ $f_i \in IE_{AB}$ $f_j \in \text{PF-PF}$ Out of requirements of merger	new class		Child: new feature A' Parent: feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{NF-CF}$ $f_i \in IE_{AB}$ $f_j \in \text{PF-PF}$ Out of requirements of merger		new class	Child: new feature B' Parent: feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{CF-NF}$ $f_i \in IE_{AB}$ $f_j \notin \text{PF-PF}$	new class		Child: new feature A' Parent: feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{NF-CF}$ $f_i \in IE_{AB}$ $f_j \notin \text{PF-PF}$		new class	Child: new feature B' Parent: feature A
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{CV-PV}$	new dimension		Child: new feature A' Parent: feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in \text{PV-CV}$		new dimension	Child: new feature B' Parent: feature A
	$IE_{AB} \in \text{volume}$ two non-connected groups of $NF \in IE_{AB}$	two new features		Child: two new features: A_1, A_2 Parent: feature B
Overlap-hiding	$IE_{AB} \in \text{CF-CF}$ Satisfying the requirements of merger	merge	Merge	Output: new feature C Hiding: feature A feature B
	$IE_{AB} = SVE_A$ $IE_{AB} \neq SVE_B$	hiding		Output: feature B Hiding: feature A
	$IE_{AB} = SVE_B$ $IE_{AB} \neq SVE_A$		Hiding	Output: feature A Hiding: feature B
	$IE_{AB} = SVE_A$ $IE_{AB} = SVE_B$	hiding		Output: feature B Hiding: feature A

	$IE_{AB} = SVE_A$ $IE_{AB} = SVE_B$		Hiding	Output: feature A Hiding: feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in CF-NF$ $f_j \in IE_{AB}$ $f_j \in PF-PF$ Satisfying the requirements of merger	merge	Merge	Output: new feature C Hiding: feature A feature B
	$IE_{AB} \in \text{volume}$ $f_i \in IE_{AB}$ $f_i \in CF-NF$ $f_j \in IE_{AB}$ $f_j \in PF-PF$ Satisfying the requirements of merger	merge	Merge	Output: new feature C Hiding: feature A feature B

4.4 Summary

In this chapter, a novel heuristic algorithm has been devised to recognise interacting features. Basic terms and concepts related to the proposed algorithm have been introduced. The algorithm for identifying the relationships between feature pairs has been described with examples. The heuristic algorithm analyses the Interacting Entity (*IE*) between each feature pair instead of the new volume created by all interacting features, making the process simpler. In addition, the proposed algorithm skips features for which no recognition is necessary, and is therefore more efficient.

Chapter 5.

ANN-based Feature Recogniser

A novel neural network-based technique for feature recognition has been developed to overcome limitations of the existing methods. This includes a novel input representation with two matrices, a suitable hierarchical net topology, a conjugate gradient training method and an output node format. This chapter discusses the ANN-based methodology proposed.

5.1 Design of neural network

The tasks of designing a neural network include:

- 1) Designing the appropriate input representation which describes features correctly and uniquely. This representation will directly influence the design of input neurons in the neural network.
- 2) Scaling each feature in a range (e.g. [0, 1]) and determining corresponding the output format. This format decides how the neural network communicates back to the environment, and therefore potentially devises the output neurons of the neural network.
- 3) Choosing an appropriate network model (such as multi-layer feedforward networks or competitive networks) and determining the topology of neural

network. This includes the kind and number of neurons, kind and number of connections between these neurons and the activation functions.

- 4) Choosing the learning method and specifying the learning parameters, e.g. learning rate.

Once the network has been designed, it has to be trained to produce the expected output vectors as a function of a predetermined pattern of input vectors [Santochi and Dini, 1996]. The training procedure in a supervised learning concerns a set of training examples forming the input and target vectors. With a pre-chosen training algorithm, the network can learn by itself. The weights will be modified step by step in order to minimise the network error. The capability of the network to obtain a low value of the error depends on several aspects such as the network architecture, training algorithm, initial values of weights and biases, set of proposed examples, number of training epochs (the term 'epoch' means a complete loop of acquisition through all the training inputs and the target vectors) [Santochi and Dini, 1996]. The following sections will introduce the proposed neural network-based techniques for feature recognition considering the above-mentioned aspects: a novel input representation with two matrices, an output node format a suitable hierarchical network topology and a conjugate gradient training method.

5.2 Input representation

As a key interface between the feature-based model and the neural network, a satisfactory input representation for the neural network has three basic characteristics:

- 1) Complete information (e.g. faces, edges and vertices) for feature recognition. It is extremely important that this representation describes features correctly and does not distort any information.
- 2) An identifiable format by the input layer of the neural network.
- 3) A unique input representation without overlaps. In other words, features belonging to different feature classes must have different input representations.

5.2.1 Attributed Adjacency Graph (AAG)

The Attributed Adjacency Graph (AAG) is a face-edge graph describing the geometry and topology of a feature pattern. Its nodes and arcs represent the faces and edges of the object's boundary respectively. The convexity information of the edge is attached to the arcs. Therefore, AAG can be defined as a graph $G = \{N, C, A\}$ [Nezis and Vosniakos, 1997], where N is the set of nodes, C is the set of connection arcs between the nodes and A is the set of connection attributes which denote the kind of connection (convex or concave). A number of neural network-based feature recognition systems use an adjacency matrix (AM) converted from the AAG as an input representation, e.g. Nezis and Vosniakos [1997]. Although

this method can recognise planar and simple curved faces correctly, it still has several problems, namely:

- 1) The representation is ambiguous. For instance, the through-slot and the through-step shown in Figure 5.1 have the same face-edge graph. It confuses the neural network and therefore the recognition output is invalid.

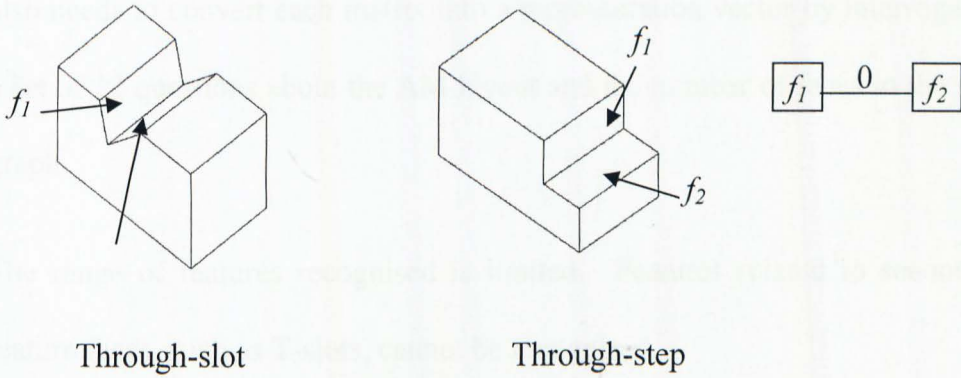


Figure 5.1 Features with the same face-edge graph

	f_1	f_2	f_3	f_4
f_1		1	0	1
f_2	1		1	1
f_3	0	1		1
f_4	1	1	1	

Matrix A

	f'_1	f'_2	f'_3	f'_4
f'_1		0	1	1
f'_2	0		1	1
f'_3	1	1		1
f'_4	1	1	1	

Matrix B

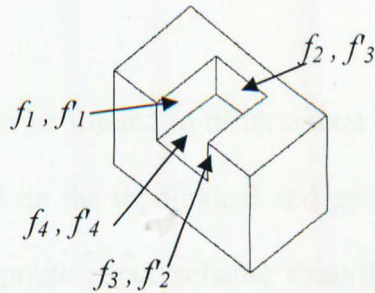


Figure 5.2 A feature with two matrices

- 2) The representation is not unique. For example, as shown in Figure 5.2, the blind slot can be represented by either Matrix A or Matrix B.
- 3) The size of matrix increases quickly as the number of faces consisting of feature increases.
- 4) It not only needs to break AAG into sub-graphs using a heuristic method but also needs to convert each matrix into a representation vector by interrogating a set of 12 questions about the AM layout and the number of faces in the sub-graph.
- 5) The range of features recognised is limited. Features related to secondary feature faces, such as T-slots, cannot be recognised.

Aiming to solve problems mentioned above, a novel input representation with two matrices is proposed in this research.

5.2.2 Proposed input representation

As shown in Figure 5.3, a neural network can be trained to differentiate between the patterns of the slot and the pocket based on the topological and geometrical information of the SVE. Thus, an appropriate input scheme describing the topological and geometrical information of a feature SVE can be used for neural network-based feature recognition. The proposed input representation works in three stages. The first stage employs the depth-first method to search for all faces in the feature SVE, builds its UndiGraph and determines the Node Sequence. The second stage defines the F-adjacency matrix. The third stage constructs a V-adjacency matrix.

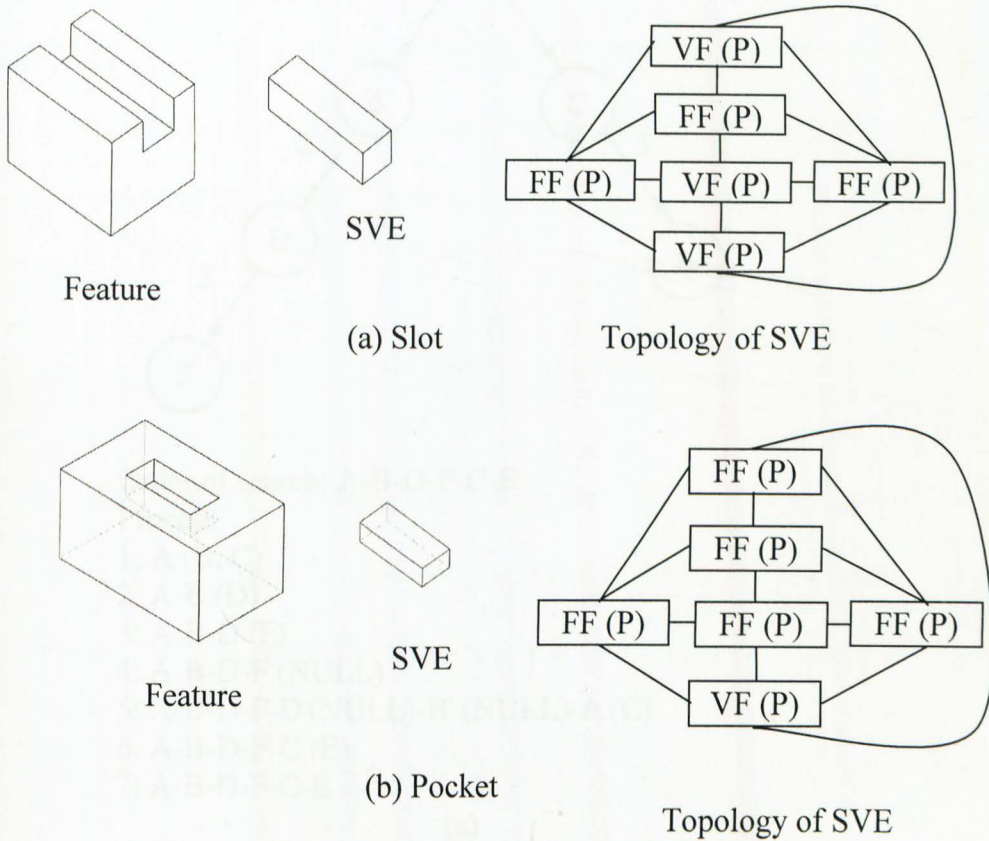
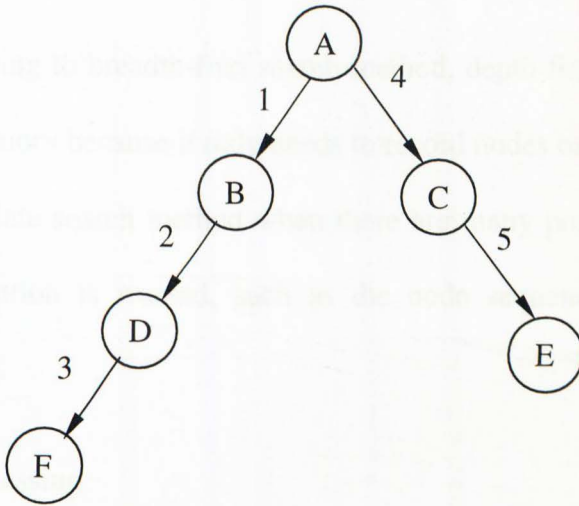


Figure 5.3 Examples of SVE, where P indicates a planar face

1) Depth-first search method

Depth-first search method is a traversal algorithm to reach all the nodes in a Graph. When possible, the algorithm always chooses an unvisited node adjacent to the current node to visit next until reaching a node that has no unvisited adjacent nodes. If all nodes adjacent to the current node have already been visited, the algorithm will backtrack to the last node that still has unvisited adjacent nodes and pick one. In other words, it always chooses to go "deeper" into the graph. The search algorithm will continue until all nodes in the graph have been visited. Figure 5.4 illustrates two examples applying the depth-first search method, with the arcs labelled in the order they are explored.

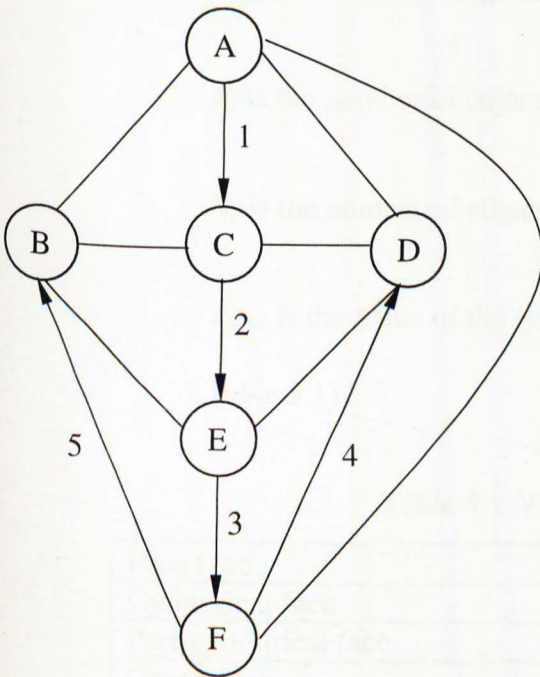


Order of search: A-B-D-F-C-E

Process

- 1: A (B, C)
- 2: A-B (D)
- 3: A-B-D (F)
- 4: A-B-D-F (NULL)
- 5: A-B-D-F-D'(NULL)-B' (NULL)-A'(C)
- 6: A-B-D-F-C (E)
- 7: A-B-D-F-C-E

(a)



Order of search: A-C-E-F-D-B

Process

- 1: A (B, C, D)
- 2: A-C (B, E, D)
- 3: A-C-E (B, F, D)
- 4: A-C-E-F (B, D)
- 5: A-C-E-F-D (NULL)
- 6: A-C-E-F-D-F' (B)
- 7: A-C-E-F-D-B

(b)

Figure 5.4 Examples of Depth-first search method

Comparing to breadth-first search method, depth-first search method requires less memory because it only needs to record nodes on the current path. It is an appropriate search method when there are many possible solutions, and only one solution is wanted, such as the node sequence in UndiGraph in this research.

2) Pre-processing

In order to transfer feature UndiGraph into an ordered adjacency list, each graph node is first assigned a priority order - Node Sequence. A Node Sequence corresponding to each face is defined as the following:

$$NS_{facei} = N_f * 10 + (6 - N_v) + T_{ftype} * 0.1 \quad (5-1)$$

where NS_{facei} is the Node Sequence of face i ;

N_f is the number of adjacent faces of face i ;

N_v is the number of adjacent virtual faces of face i ;

T_{ftype} is the value of the type of face i (the value allocated is shown in Table 5.1).

Table 5.1 Value of face type

Face type	Value
Cylindrical face	1
Part-cylindrical face	2
Conical face	3
Part-conical face	4
Semi-spherical face	5
Planar face	6
Linear-group	7
Circular-group	8

According to the Node Sequence and the algorithm of the depth-first search, a priority order of faces is determined by an adapted depth-first traversal method. The process is detailed below.

Step 0: Let

- OAL be an Ordered Adjacency List, which is initialized as Null.
- $USet$ be a set of faces that have not been visited yet.

Step 1: Choose a starting face, $F=face_i$, where

- $face_i$ is an un-visited face in the feature, $face_i \in USet$;
- $face_i$ has the lowest value of Node Sequence among the un-visited faces,

$$NS_{face_i} \leq NS_{face_k \in \{face_k \mid face_k \in USet\}}$$

If $F = \text{Null}$, then go to end. Else, go to Step 2.

Step 2: Add $face_i$ to OAL and delete $face_i$ from $USet$.

Step 3: If $USet = \emptyset$, then go to end. Else, go to the next step.

Step 4: Choose a face $face_j$ where

- $face_j$ is an un-visited face in the feature, $face_j \in USet$.
- $face_j$ is adjacent to $face_i$, $face_j \in \{face_k \mid face_k \text{ is adjacent to } face_i\}$.
- $face_j$ has the lowest value of Node Sequence among the un-visited adjacent

$$\text{faces to } face_i, NS_{face_j} \leq NS_{face_k \in \{face_k \mid face_k \text{ is adjacent to } face_i \ \& \ face_k \in USet\}}$$

If there is no face satisfying the above conditions, go to Step 1.

If there exists only one satisfactory face, then $F=face_j$ and $face_i = face_j$; Go to Step 2.

If there are two or more faces satisfying the above requirements, define a new set, FS and add all faces satisfying the conditions to FS . Then go to Step 5.

Step 5: Choose the face whose angle with F is the smallest. That is

- $face_j \in FS$
- $face_j$ has the smallest angle, $angle_{(face_j, face_i)} \leq angle_{(face_k, face_i)} \forall face_k \in FS$,
where $angle_{(face_j, face_i)}$ is the angle between $face_j$ and $face_i$.

Then $F=face_j$ and $face_j=face_i$, Go to Step 2.

Figure 5.5 summarises the above steps and Figure 5.6 shows an example of pre-processing.

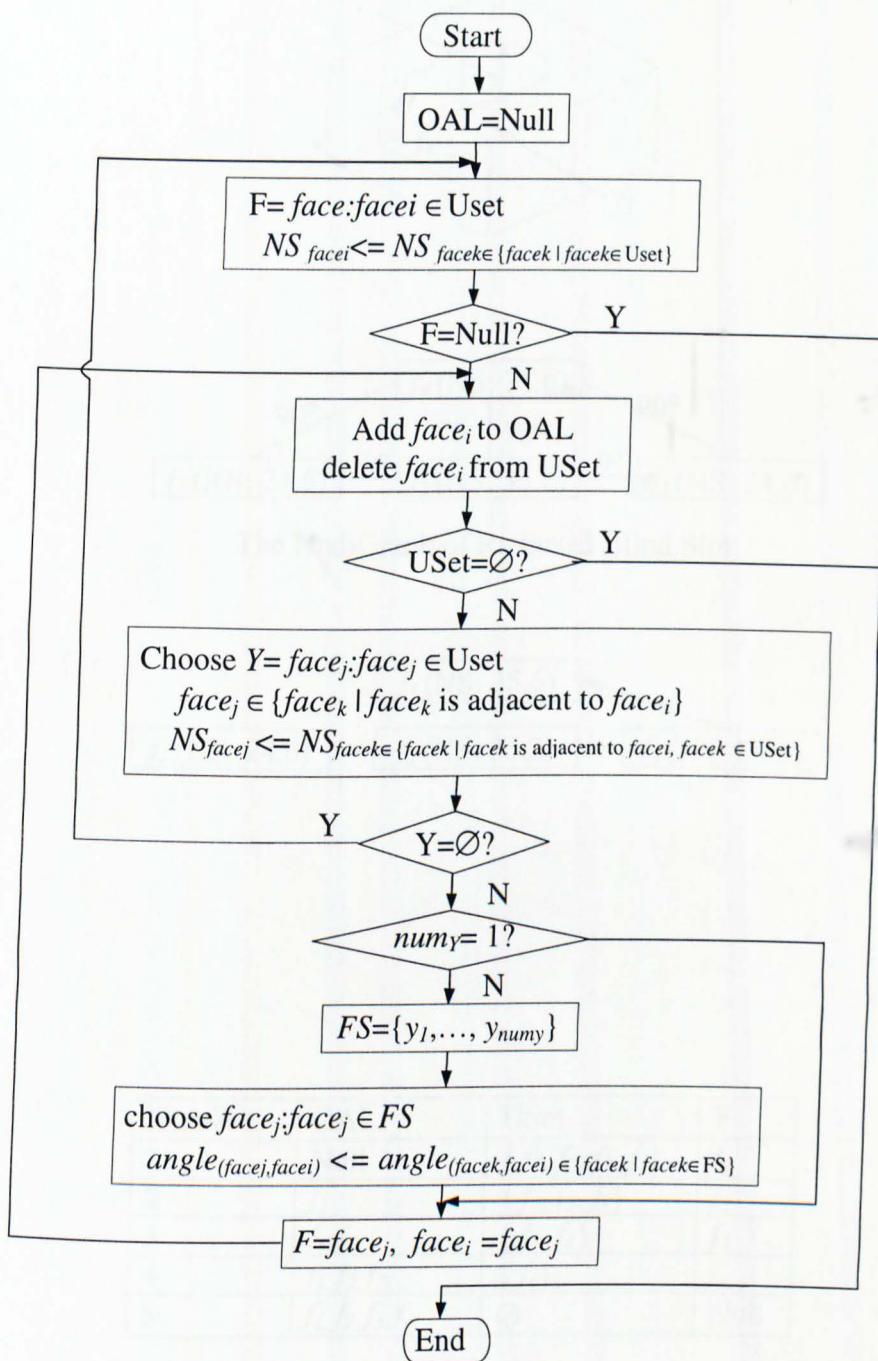
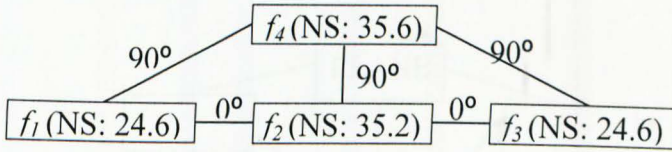
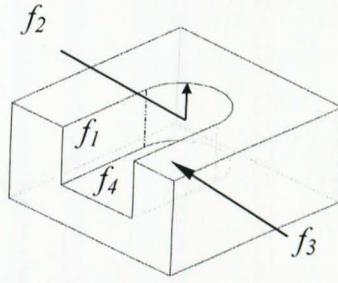
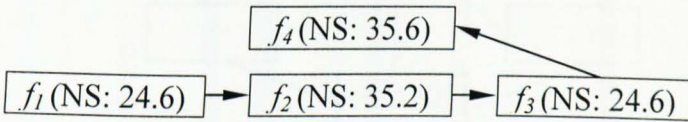


Figure 5.5 Pre-process steps



The UndiGraph of Radiused Blind Slot



Number	OAL	USet	F
1	Null	$\{f_1, f_2, f_3, f_4\}$	f_1
2	f_1	$\{f_2, f_3, f_4\}$	f_2
3	f_1, f_2	$\{f_3, f_4\}$	f_3
4	f_1, f_2, f_3	$\{f_4\}$	f_4
5	f_1, f_2, f_3, f_4	\emptyset	Null

Figure 5.6 Pre-processing of Radiused Blind Slot

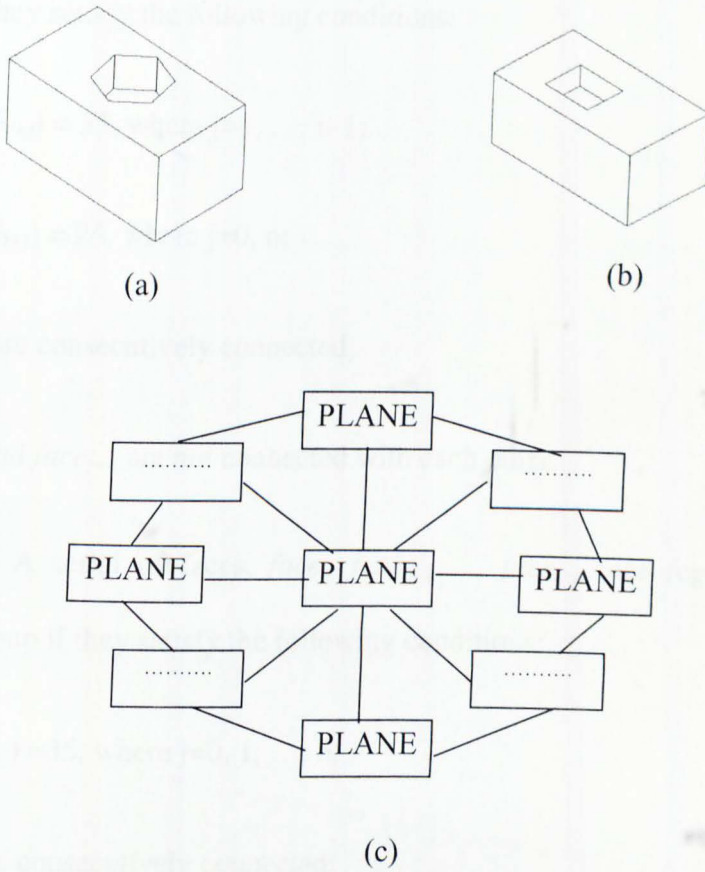


Figure 5.7 Simplification of topology

With the number of faces increased, the size of matrix will become quite large. For example, the pocket shown in Figure 5.7 (a) consists of seven faces and the size of the adjacency matrix will be 7×7 . In practical cases, the size of the matrices can be reasonably decreased. As shown in Figure 5.7 (a), the topological information is similar to the pocket in Figure 5.7 (b) and can be described as the graph shown in Figure 5.7 (c). If the number of faces in the OAL is larger than 5, the OAL should be simplified. The rules for simplification are described below.

Rule 1: A series of faces, $face_i, face_{i+1}, \dots, face_{i+n}$, are regarded as a Linear Group if they satisfy the following conditions:

- $int(NS_{i+j}) = 35$, where $j=1, \dots, n-1$;
- $int(NS_{i+j}) = 24$, where $j=0, n$;
- They are consecutively connected;
- $face_i$ and $face_{i+n}$ are not connected with each other.

Rule 2: A serial of faces, $face_i, face_{i+1}, \dots, face_{i+n}$, are regarded as a Circular Group if they satisfy the following conditions:

- $int(NS_{i+j}) = 35$, where $j=0, 1, \dots, n$;
- They are consecutively connected;
- $face_i$ and $face_{i+n}$ are connected with each other.

3) F-adjacency Matrix

F-adjacency matrix is used to recognise five primitive features: round hole, conical hole, general hole, slot/step, pocket. It is defined as $I_F = [a_{ij}]_{i \times j}$, where $1 \leq i \leq 5$ and $1 \leq j \leq 5$.

$$I_F = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

The layout of I_F is ergonomically designed to have a one-to-one correspondence between the feature pattern and the input matrix. The middle elements of I_F , i.e. a_{ii} , show the type of the i th face, $face_i$ (e.g. 6 for a planar face). Table 5.1 denotes the values for various face types. Other elements of I_F (a_{ij} , where $i \neq j$) indicate the connection between the i th and j th faces of the object. A numerical value between 0 to 9 is allocated according to the relationship between the two faces. The values are given in Figure 5.8.

The layout presentation of I_F is symmetrical so that the input format consists of 15 nodes, $a_{11}, a_{12}, \dots, a_{15}, a_{22}, a_{23}, \dots, a_{25}, \dots, a_{55}$. Some F-adjacent Matrices of features are presented in Table 5.2.

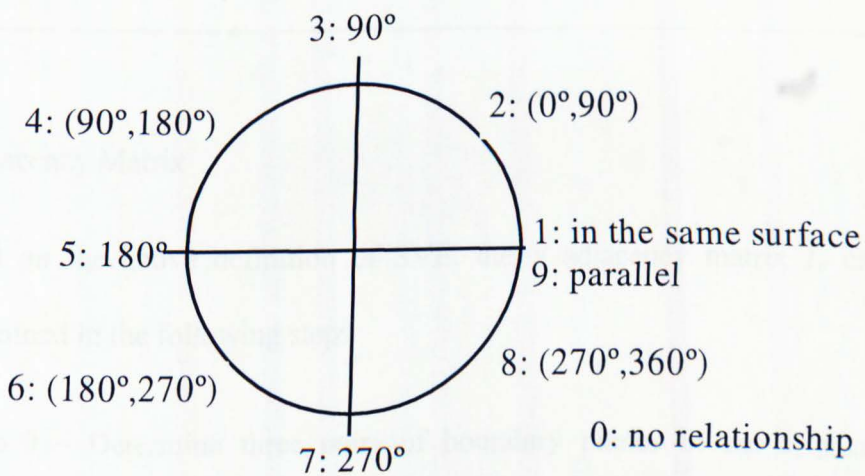
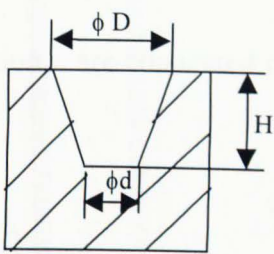
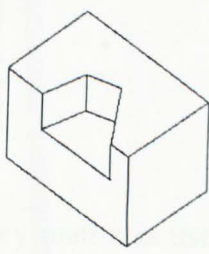


Figure 5.8 Values of relationship between two faces

Table 5.2 Examples of F-adjacency Matrix

Feature	F-adjacency matrix
Blind conical hole	$3\ 4\ 0\ 0\ 6\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$ $I_F = \begin{bmatrix} 3 & 4 & 0 & 0 & 0 \\ & 6 & 0 & 0 & 0 \\ & & 0 & 0 & 0 \\ & & & 0 & 0 \\ & & & & 0 \\ & & & & & 0 \\ & & & & & & 0 \\ & & & & & & & 0 \\ & & & & & & & & 0 \\ & & & & & & & & & 0 \\ & & & & & & & & & & 0 \\ & & & & & & & & & & & 0 \\ & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & & & 0 \end{bmatrix}$ 
Open pocket	$6\ 3\ 0\ 4\ 0\ 6\ 3\ 3\ 0\ 6\ 4\ 0\ 6\ 0\ 0$ $I_F = \begin{bmatrix} 6 & 3 & 0 & 4 & 0 \\ & 6 & 3 & 3 & 0 \\ & & 6 & 4 & 0 \\ & & & 6 & 0 \\ & & & & 0 \\ & & & & & 0 \\ & & & & & & 0 \\ & & & & & & & 0 \\ & & & & & & & & 0 \\ & & & & & & & & & 0 \\ & & & & & & & & & & 0 \\ & & & & & & & & & & & 0 \\ & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & & 0 \\ & & & & & & & & & & & & & & & 0 \end{bmatrix}$ 

4) V-adjacency Matrix

Based on the above definition of SVE, the V-adjacency matrix I_V can be determined in the following steps.

Step 1: Determine three pairs of boundary planes in the x , y and z directions, which can be represented as $+x$, $-x$, $+y$, $-y$, $+z$ and $-z$.

Step 2: Define the SVE for the given feature, which is completely enclosed based on the above six directions.

Step 3: Attach the attributes of FF/VF to all faces in the SVE.

Step 4: Define a 6×6 matrix, I_V showing the relationships between VF faces in the SVE. The middle element, b_{ii} , show whether there is a VF in the

corresponding direction. If in the i th direction (e.g. $+x$), the given SVE exists a VF face, then $b_{ii} = 1$; if not, $b_{ii} = 0$. The elements, b_{ij} ($i \neq j$), describe whether the two VFs, corresponding to direction i and direction j , are connected or not (i.e. 1 or 0).

$$I_V = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} \end{bmatrix}$$

Similarly, the symmetric characteristic of V-adjacency matrix is used to the simplification of its input. A vector consisting of 21 codes is input to the neural network. That is, $b_{11}, b_{12}, \dots, b_{16}, b_{22}, b_{23}, \dots, b_{26}, \dots, b_{66}$. Two examples are provided in Table 5.3.

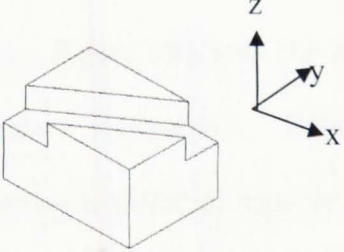
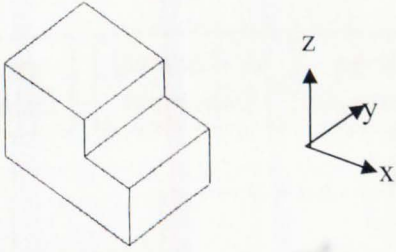
5.3 Output format

The output of an ANN is the result of many operations with the input and weights.

Commonly, a good output format should have the following characteristics:

- 1) Representation scheme: It is essential that the output describes the results clearly and correctly as expected. In this research, the final result is a feature class that the given feature belongs to.

Table 5.3 Examples of V-adjacency Matrix

Feature	V-adjacency matrix
Slot	<p>1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0</p> $I_V = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ & 1 & 0 & 1 & 1 & 0 \\ & & 1 & 0 & 1 & 0 \\ & & & 1 & 1 & 0 \\ & & & & 1 & 0 \\ & & & & & 0 \end{bmatrix}$ 
Step	<p>1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0</p> $I_V = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 1 & 0 \\ & & & 1 & 1 & 0 \\ & & & & 1 & 0 \\ & & & & & 0 \end{bmatrix}$ 

- 2) Appropriate format: Similar to the input format, the output is designed as a nodal value in the format of a vector. Because of the proposed hierarchical architecture of ANN-based feature recognition (described further in section 5.4), the number of feature classes to be recognised is small. Thus, it is possible that each output neuron represents a certain feature class.
- 3) Activation method: For feature recognition, it is not practicable to activate two classes at the same time. Therefore, only one of the output neurons will be activated (i.e. its value will be greater than the threshold value, 0.5). If one or more output neurons are activated, the pattern presented to the network does not belong to a known class, while the class with the greatest value is considered.

5.4 The topology of neural network

As mentioned before, the proposed feature classification is hierarchical, where the sub-class is regarded as an instance of its parent class. Figure 5.9 shows the three-level hierarchical architecture of ANN-based feature recognition. Different topologies and learning methods are designed according to different requirements of the three levels. The characteristics of the feature recognition are:

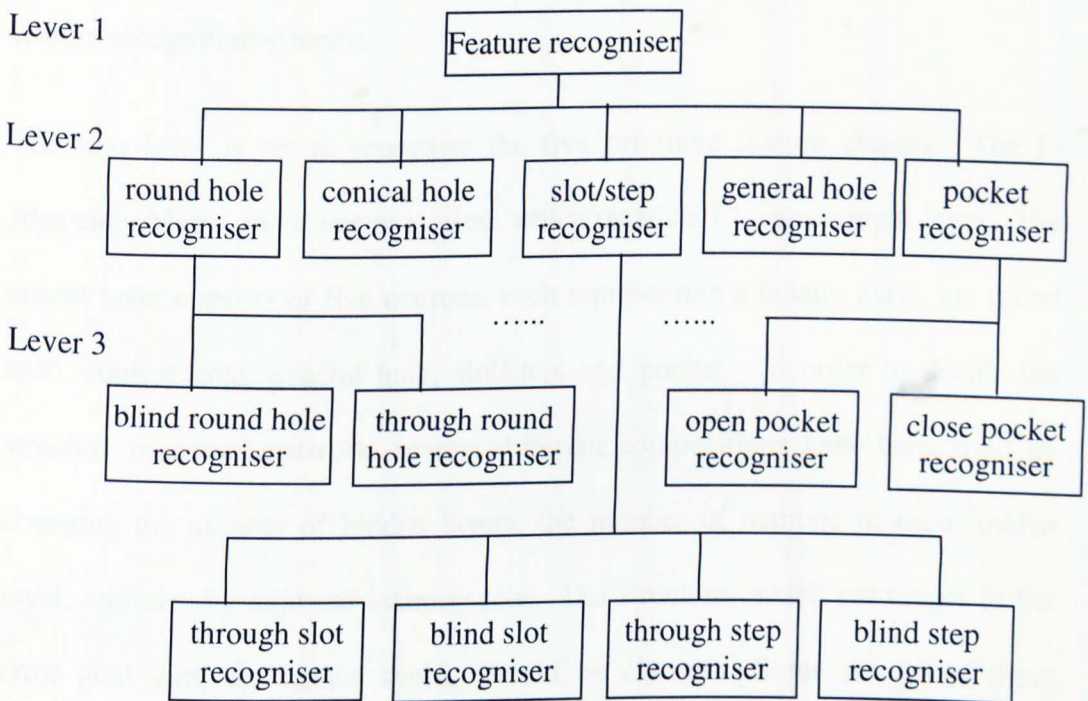


Figure 5.9 Hierarchical neural networks system

- 1) Feature recognition is mainly based on the 3D geometrical and topological information of features, not on the input probability. In addition, it is possible to collect enough samples including inputs and targets due to various machining features existing in practical industry.

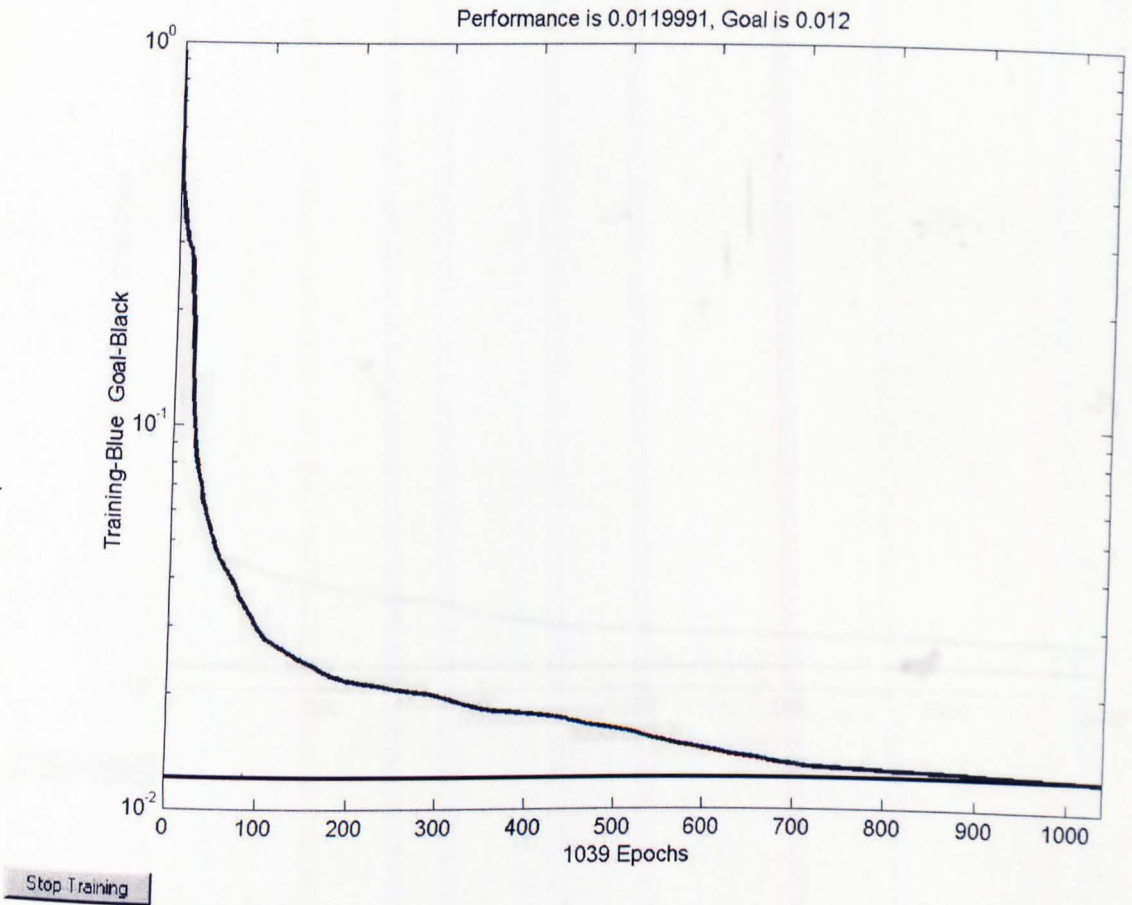
- 2) Machining feature recognition is a complicated process, for which the entire information including both geometric and topologic information of features recognised needs to be input.

According to the above requirements, comparing to single-layer feedforward network, competitive network and recurrent network, multi-layer feedforward network are more suitable for this research. They are also the choices for most feature recognition systems.

The first level is set to recognise the five primitive feature classes. The F-adjacency Matrix input vector is used, which means a 12-neuron input layer. The output layer consists of five neurons, each representing a feature class, i.e. round hole, conical hole, general hole, slot/step, and pocket. In order to decide the structure of neural network, several different compositions have been tried by changing the number of hidden layers, the number of neurons in each hidden layer, and also by adjusted learning rate. The structure, which converges to the error goal with the fastest speed, should be chosen. Some results of these experiments are given in Figure 5.10 (a-c) and appendix B. Based on these results, the network consisting of three layers with a hidden layer of 17 neurons has proved to be the most appropriate structure (Figure 5.11).

The second and third levels are used for further recognition based on the first level for computer aided process planning (CAPP) applications. The structures of neural network for the second level recognition are designed following the same steps. For example, with various experiments, the slot/step classifier is designed

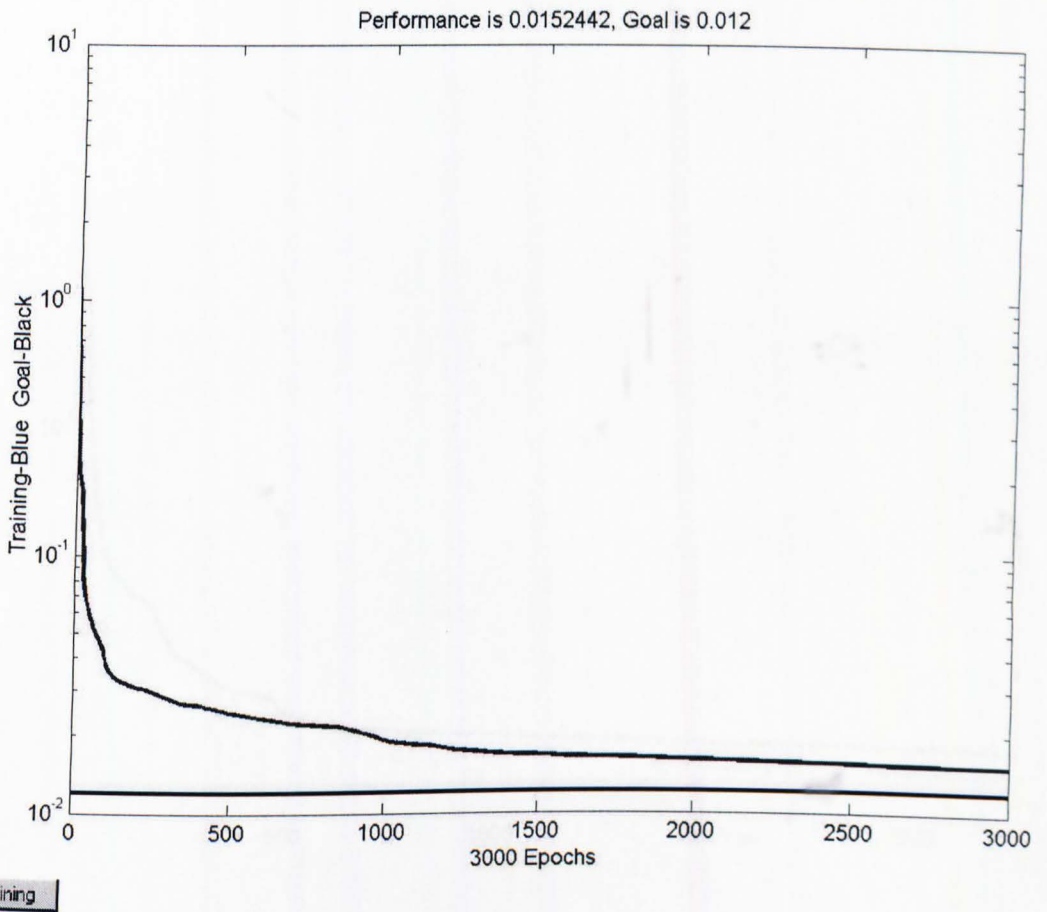
using a V-Matrix as input, which consists of three layers, an input layer of 21 neurons, a hidden layer of 6 neurons and an output layer of 4 neurons.



(a)

Figure 5.10 The training process of neural networks
(a) structure of 15-17-5

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}}$$

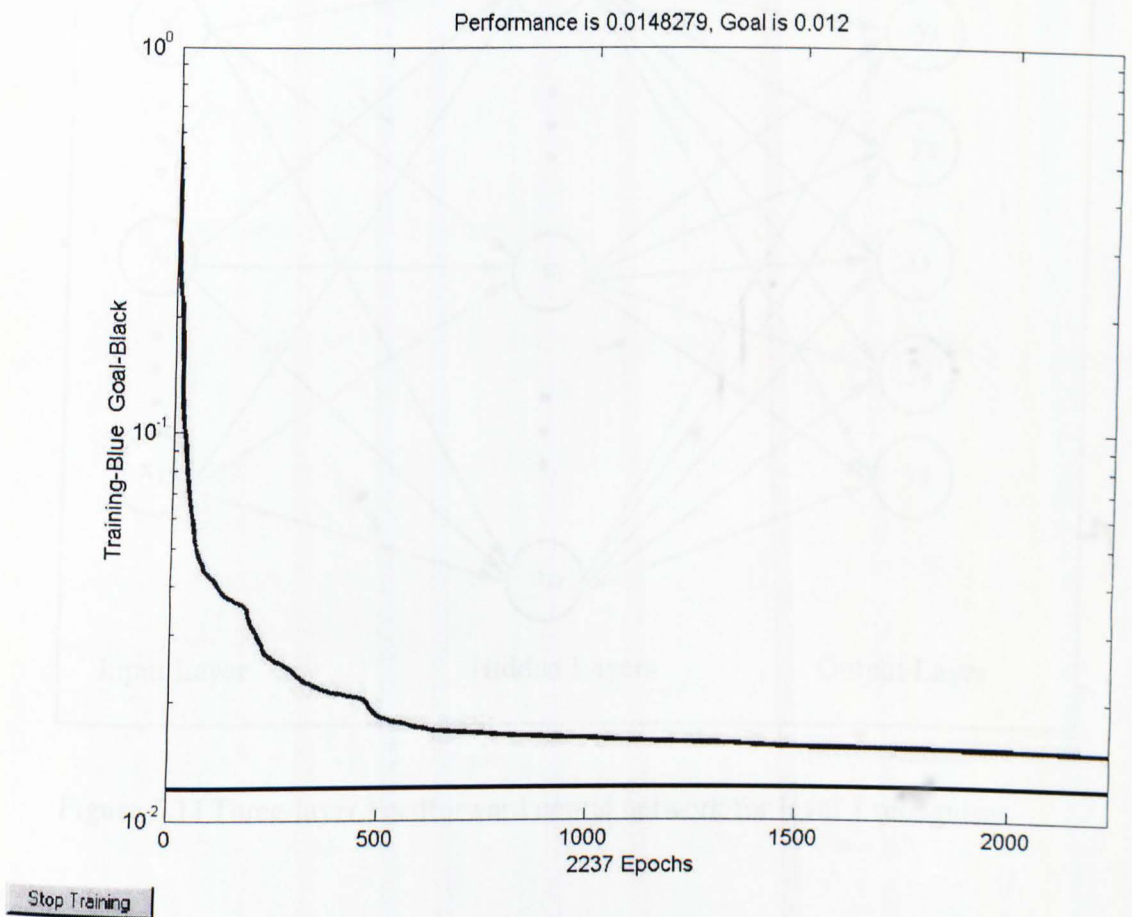


(b)

Figure 5.10 The training process of neural networks (continued)

(b) structure of 15-16-5

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}}$$



(c)

Figure 5.10 The training process of neural networks (continued)

(c) structure of 15-15-5

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}}$$

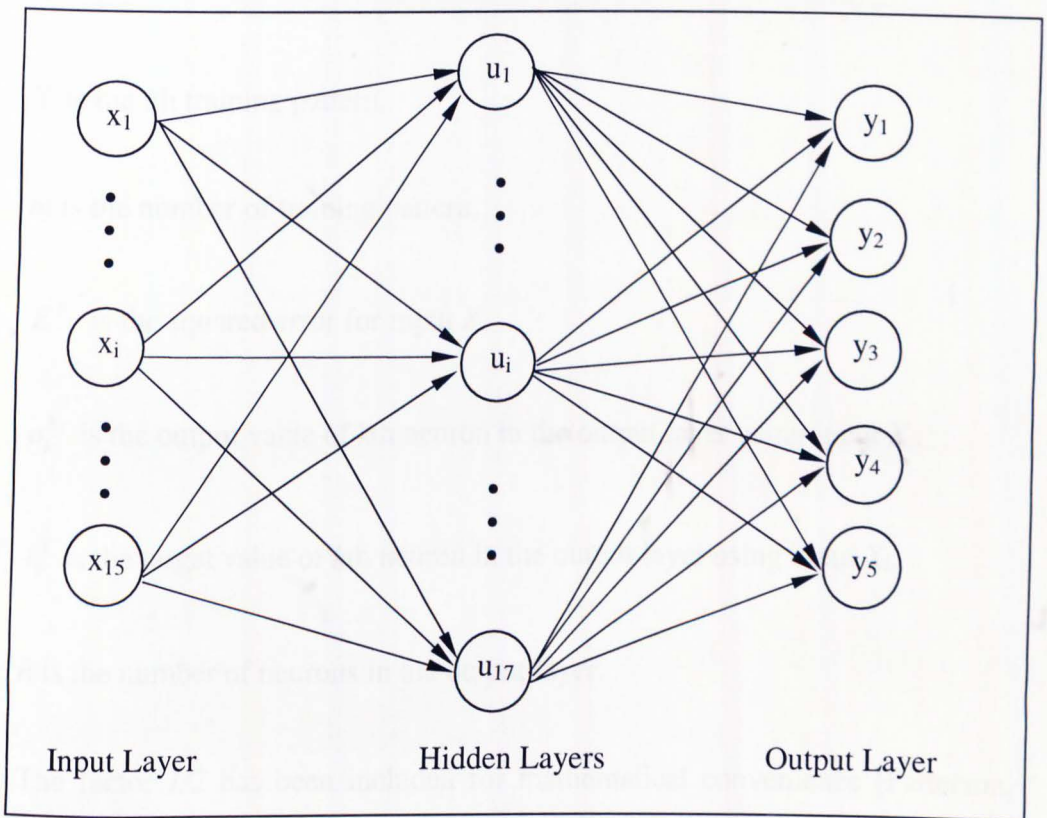


Figure 5.11 Three-layer Feedforward neural network for level 1 recogniser

5.5 Error function

The error function E , can be defined in different ways, for example, the mean square error, the absolute error, etc. [Patterson, 1996]. This research uses the mean square error, which equals the mean of the squares of the deviations from target. It can be indicated as

$$E = \frac{1}{m} \sum_{i=1}^m E^{X_i} \quad (5-2)$$

$$E^{X_i} = \frac{1}{2} \sum_{k=1}^n (o_k^{X_i} - t_k^{X_i})^2 \quad (5-3)$$

where

X_i is the i th training pattern.

m is the number of training pattern.

E^{X_i} is the squared error for input X_i .

$o_k^{X_i}$ is the output value of k th neuron in the output layer using input X_i .

$t_k^{X_i}$ is the target value of k th neuron in the output layer using input X_i .

n is the number of neurons in the output layer.

The factor $1/2$ has been included for mathematical convenience [Patterson, 1996]. The mean square error penalises large deviations and provides a differentiable, decreasing function of the difference between the computed and desired outputs [Patterson, 1996]. It is one of the most commonly used error measures in back propagation neural networks. According to the results of experiments, the mean square error is set to 0.012.

5.6 Training of ANN

Before the process of recognition, the neurons in neural network have to be trained with training examples. One of the supervised training methods, commonly used in current feature recognition systems, such as Chen and Lee [1998], Nezis and Vosniakos [1997], Zulkifli and Meeran [1999], is the back propagation algorithm. The basic BP algorithm adjusts the weights in the steepest

descent direction (negative of the gradient) [Demuth and Beale, 2000]. Although this direction makes the performance function (Error function E) decrease most rapidly, it does not necessarily produce the fastest convergence. Alternative approaches, known as conjugate gradient algorithms, make a search along conjugate directions, which produces generally faster convergence than in the steepest directions. A set of mutually conjugate directions can be achieved through the following steps.

Step 0: An initial weight vector ($W^{(0)}$) is chosen randomly.

Step 1: The steepest descent direction (d_0) is selected on the first iteration, which is the negative of the gradient (g_0),

$$d_0 = -g_0 \quad (5-4)$$

$$g_0 = \nabla E(W^{(0)}) \quad (5-5)$$

where $E(W)$ is the error function made up from the outputs of all the input patterns, and $\nabla E(W)$ is the the negative of the gradient vector at W .

Step 2: The weights are updated by an optimal distance (called learning rate, α_k) along the current search direction,

$$W^{(k+1)} = W^{(k)} + \alpha_k d_k. \quad (5-6)$$

Here, α_k is determined using a line search method proposed by Charalambous [1992], which minimises the error function along the current search direction.

Assuming a function $\psi(\alpha)$ is defined as

$$\psi(\alpha_k) = E(W^{(k)} + \alpha d_k) \quad (5-7)$$

and it satisfies

$$\psi(\alpha_k) < \psi(0) \quad (5-8)$$

$$\psi'(\alpha_k) = d\psi(\alpha) / d\alpha \quad (5-9)$$

Based on the above definition, α_k must satisfy two requirements shown below:

Requirement I:

$$\psi(\alpha_k) \leq \psi(0) + \mu\psi'(0) \quad (5-10)$$

where μ is a small number less than 0.5.

Requirement II:

$$|\psi'(\alpha_k)| \leq -\sigma\psi'(0) \quad (5-11)$$

where $\sigma \in (0,1)$ and $\sigma \leq \mu$.

Then, a value of α_k is determined by a line search algorithm, which is based on the cubic interpolation. The algorithm is described below.

Suppose that the initial point P_i , $\alpha = \alpha_1(P_i)$ and the positive step size taken is $\bar{\alpha}$:

$$\bar{\alpha} = \max(\alpha_p, \alpha_q^*) \quad (5-12)$$

where

$$\alpha_q^* = -\frac{2\Delta E}{s_0} \quad (5-13)$$

$$s_0 = d_k^T \nabla E(W_k) \tag{5-14}$$

α_p is the value obtained at the previous iteration.

ΔE is the last value of E . At the beginning, ΔE must be user-supplied.

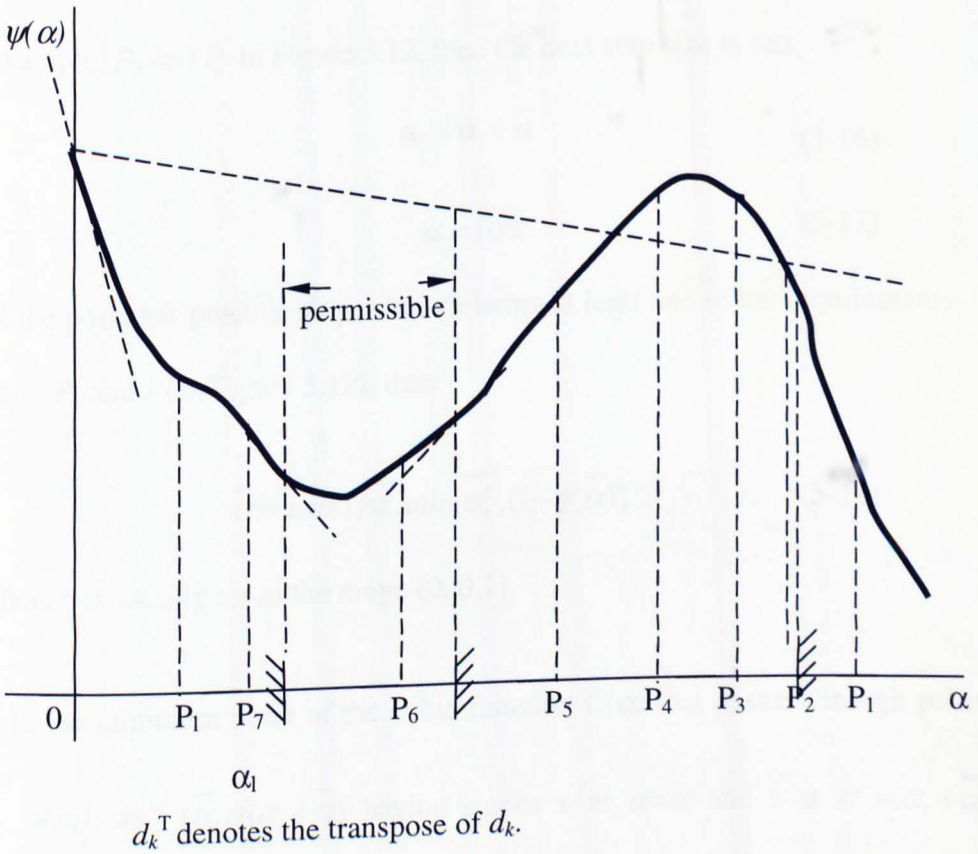


Figure 5.12 Illustration of the line search algorithm [Charalambous, 1992]

Then, the step size for the next point is produced based on the following rules:

- 1) If it satisfies both requirements (i.e. P_6 in Figure 5.12), the line search will stop.

- 2) If the point has negative slope and the function value is greater or equal to $\psi(\alpha_1)$, such as P_2 and P_3 in Figure 5.12, then a large step size for the next repeat is taken. That is

$$\bar{\alpha} = 0.1\bar{\alpha} \quad (5-15)$$

- 3) If the point has negative slope and the function value is less than $\psi(\alpha_1)$, for example, P_1 and P_7 in Figure 5.12, then the next step size is set:

$$\alpha_l = \alpha_l + \bar{\alpha} \quad (5-16)$$

$$\bar{\alpha} = 10\bar{\alpha} \quad (5-17)$$

- 4) If the point has positive slopes but violating at least one of the requirements (e.g. P_4 and P_5 in Figure 5.12), then

$$\bar{\alpha} = \max\{\gamma\bar{\alpha}, \min(\bar{\alpha}_c^*, (1-\gamma)\bar{\alpha})\} \quad (5-18)$$

Where γ is usually set in the range (0, 0.1)

$\bar{\alpha}_c^*$ is the minimum point of the cubic function $C(\alpha)$ that passes through points $(0, \psi(\alpha_1))$ and $(\bar{\alpha}, \psi(\alpha_1 + \bar{\alpha}))$ having slopes s_l at $\alpha = \alpha_l$ and s at $\alpha = \alpha_l + \bar{\alpha}$, where.

$$s_l = d_k^T \nabla E(W^{(k)} + \alpha_l d_k) \quad (5-19)$$

$$s = d_k^T \nabla E(W^{(k)} + (\alpha_l + \bar{\alpha}) d_k) \quad (5-20)$$

Step 3: The stopping criterion (performance goal to satisfy the error set) is examined. If it is satisfied, the training stops; otherwise proceeds to the next step.

Step 4: The new gradient vector of performance (g_{k+1}) is evaluated, which is orthogonal to the previous search direction,

$$d_k^T g_{k+1} = 0 \quad (5-21)$$

Step 5: Each successive direction (d_{k+1}) is chosen as a linear function of the current gradient and the previous search direction (d_k),

$$d_{k+1} = -g_{k+1} + \beta_k d_k \quad (5-22)$$

Step 6: Set $k=k+1$, go to Step 2.

Two search functions have been tested to find the coefficient β_k in order to determine the direction to minimise the performance function; they are:

1) Fletcher-Reeves

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \quad (5-23)$$

2) Polak-Ribiere

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}} \quad (5-24)$$

It can also be demonstrated that the Polak-Ribiere form provides slightly better results than the other expressions because it gives a small value for β_k . Figure 5.10 (a) and Figure 5.13 provides the results of experiments comparing Polak-Ribiere form with Fletcher-Reeves form.

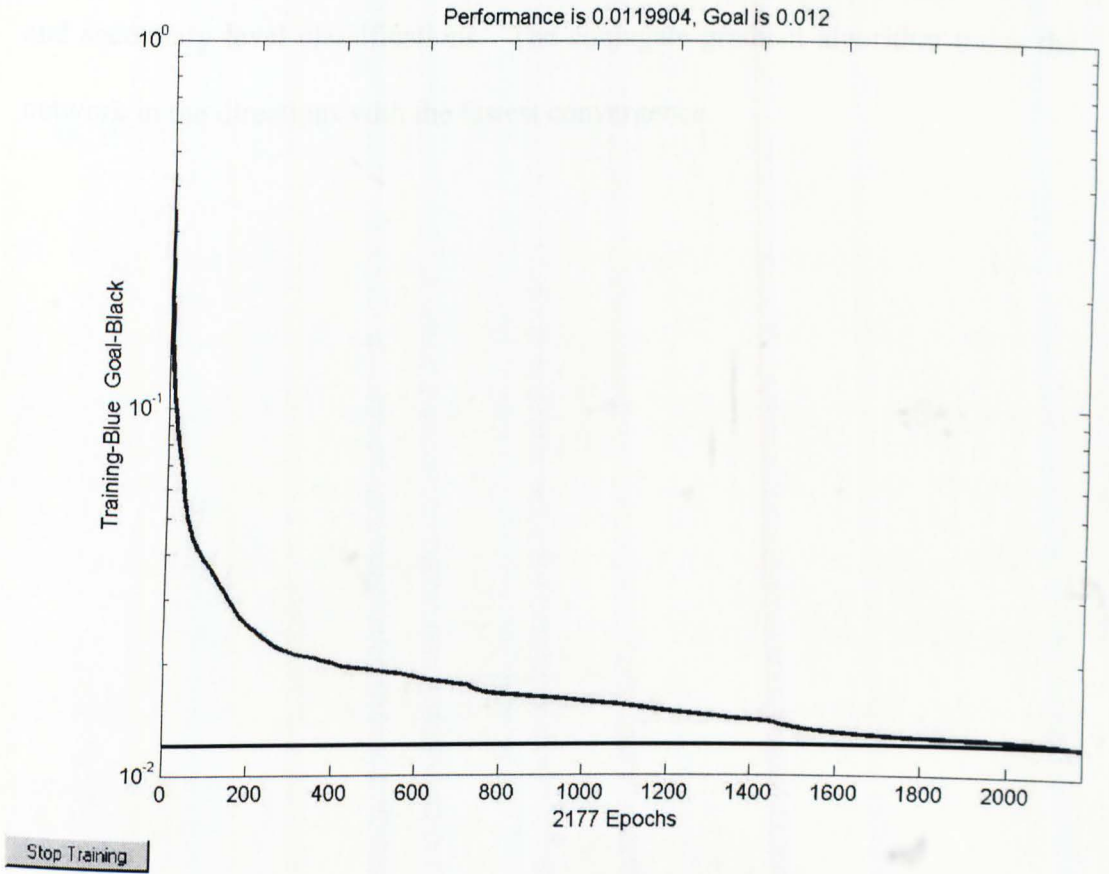


Figure 5.13 The training process of neural networks (continue)

structure of 15-17-5

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

5.7 Summary

This chapter has presented new techniques that use neural networks to improve the capability of current feature recognition methods. They are different in several respects. F-adjacent and V-adjacent matrices not only capture the topological information uniquely and clearly but also describe the parallel relationships which previous work cannot provide. The representation is suitable

for the hierarchical feature classification and has been successful for the first level and secondary level classifications. The conjugate gradient algorithm trains the network in the directions with the fastest convergence.

5.1. Requirements of CAPP

In order to overcome the weaknesses of existing CAPP systems, one may wish to consider the level of automation, suitable knowledge representation, adaptability, flexibility, integration and efficiency.

1) Level of automation: Activities to process planning should be fully carried out automatically. However, user intervention is still needed, such as reviewing the process plan or adding alternatives provided by system, etc.

2) Suitable knowledge representation: It is difficult to transfer experience of many years of manufacturing activities in a knowledge base which can be

Chapter 6.

Compute Aided Process Planning (CAPP)

In the production preparations, the first and most important task is making a process planning, which provides information to the manufacturing workers on how to produce the designed products [Zhao and Wu, 1999]. With the growing trend towards quick response to global market changing and flexible automation production, fast and flexible generation of process plans has become essential in the new manufacturing systems. This chapter presents the work on generative CAPP for prismatic parts incorporating features technology, artificial intelligence techniques and fuzzy evaluation.

6.1. Requirements of CAPP

In order to overcome the weaknesses of existing CAPP systems, there are six aspects to consider: the level of automation, suitable knowledge representation, adaptability, flexibility, integration and efficiency.

- 1) Level of automation: Activities in process planning should be ideally carried out automatically. However, user intervention is still needed, such as improving the process plan or editing alternatives provided by system, etc.
- 2) Suitable knowledge representation: It is difficult to formalise experience of many years of manufacturing activities in a knowledge base which can be

modified and updated. Database structure and knowledge definitions are two major factors for a suitable knowledge representation.

- 3) **Adaptability:** On the one hand, each company has its own manufacturing environment, e.g. products, planning rules, manufacturing resources, standards and documentation, and a CAPP system should be able to adapt to different user requirements. On the other hand, the rapid change in industry has an impact on the planning and requires adaptability in CAPP. Therefore, it is necessary for the CAPP system to have an open and flexible architecture and a continually updated knowledge base to meet the current and future needs.

6.2. Architecture of CAPP

This research aims to integrate CAD and CAM through CAPP consisting of four steps. First, each feature input from CAD for each feature generates its corresponding machining operations, including the operation type, machine tool and cutting tool, etc. It proceeds to determine the precedence relationships among the features considering various constraints for design and manufacturing. An algorithm is then developed to group machining operations. Finally, the sequence of operation groups is obtained with a genetic algorithm. To fulfil the requirements and the steps mentioned above, an architecture of a CAPP system for prismatic components is proposed, which is shown in Figure 6.1. A brief description of each sub-module is given below:

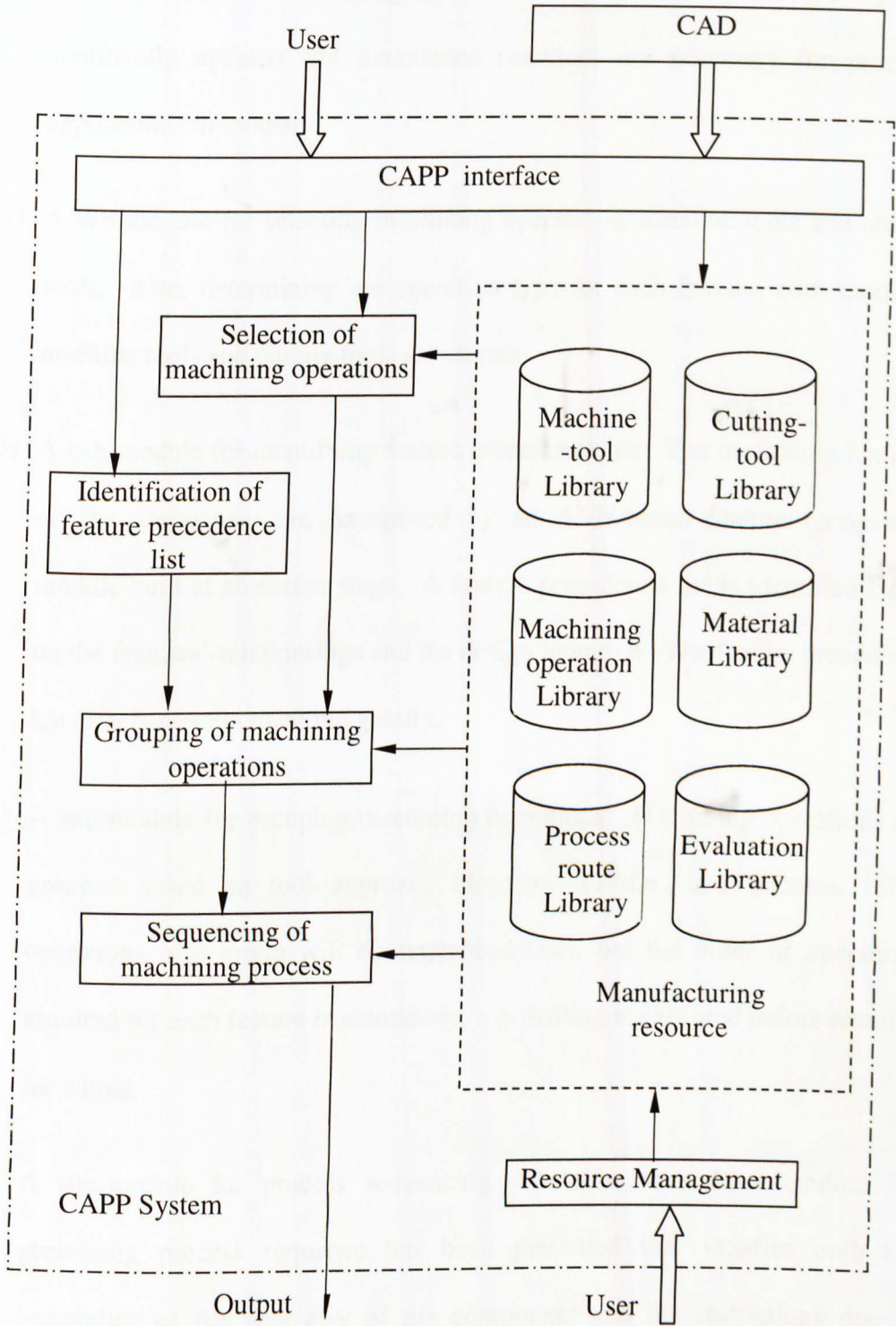


Figure 6.1 The proposed architecture of CAPP

- 1) A sub-module for managing resources. As a core of a CAPP system, continually updated and maintained resources are necessary for practical applications in industry.
- 2) A sub-module for selecting machining operations, machine tools and cutting tools. After determining the operation type for each feature, corresponding machine tools and cutting tools are chosen.
- 3) A sub-module for identifying feature precedence list. The machining features on the component are recognised by an ANN-based feature recognition module built at an earlier stage. A feature precedence list is identified based on the features' relationships and the design intention. The feature precedence list affects process planning greatly.
- 4) A sub-module for grouping machining operations. Machining operations are grouped based on tool approach directions (TADs) and features. The operations in a group will be sequenced later, but the order of operations required for each feature is mandatory, e.g. drilling is executed before reaming for a hole.
- 5) A sub-module for process sequencing. In this research, a method for generating process sequence has been presented that satisfies both the constraints of the geometry of the component and the restrictions due to manufacturing rules, while at the same time minimising the component machining cost and time. The problem has been formulated within a Genetic Algorithm (GA) framework. A population of feasible solutions (process sequences) is generated randomly by a precedence constraint initial algorithm.

Then, the population of feasible solutions is bred using selection, crossover and mutation operators. The solution with the best fitness is considered as the final result. A particular calculation for the fitness function has been developed for the proposed GA, which include evaluating degree of satisfactory of process sequence rules using the analytical hierarchy process (AHP) which is a technique for fuzzy analysis, approximated calculations for manufacturing cost and time, and an intelligent neural network for allocating weights based on fuzzy evaluation of features. In the next chapter, the sub-module is described further.

All of these sub-modules communicate with each other through a relational database. The following section describes four key issues relevant to the proposed CAPP: resource management, selection of machining operations, identification of feature precedence list and grouping machining operations.

6.3. Resource management

Resource management maintains all information in an integrated database about resources required for process planning. A suitable database needs to be designed, which requires a large amount of previous work including analysis, formalisation and representation of various manufacturing parameters and constraints, expert knowledge and experiences. Based on the data investigated from two companies, a manufacturing database is defined for this research, which contains above information with six libraries:

1) Machine-tool library

It stores all machine tools available in the given industrial environment. Each machining tool is attached with several technological parameters that must be considered during planning, such as tolerances, spindle rigidity, maximum traverse of X, Y and Z axis, and cost.

2) Cutting-tool library

The cutting-tool library stores and provides the user with the information of cutting tools available for process planning. Cutting tool parameters consists of two types: common parameters for all cutting tools, such as cost, maximum cutting speed, maximum depth of cut, material, tool rigidity and accuracy; and specific parameters, for example, face milling concerns corner angle, axial-rake angle, radial-rake angle, true-rake angle and inclination angle while peripheral milling includes cutter diameter, cutter teeth, flutes, relief angle and the width of land.

3) Material library

It includes all the materials for both components and cutting tools, material properties (e.g. rigidity, intensity and toughness), shape and size of raw materials.

4) Machining operation library

The machining operation library includes the information for all machining operations. Each machining operation contains not only the type of operation,

but also corresponding machine tool and cutting tool. In addition, three aspects of knowledge are concerned:

- capabilities, e.g. the achievable roughness
- constraints, for example, the maximum and minimum machining length
- machining cost and time

5) Process route library

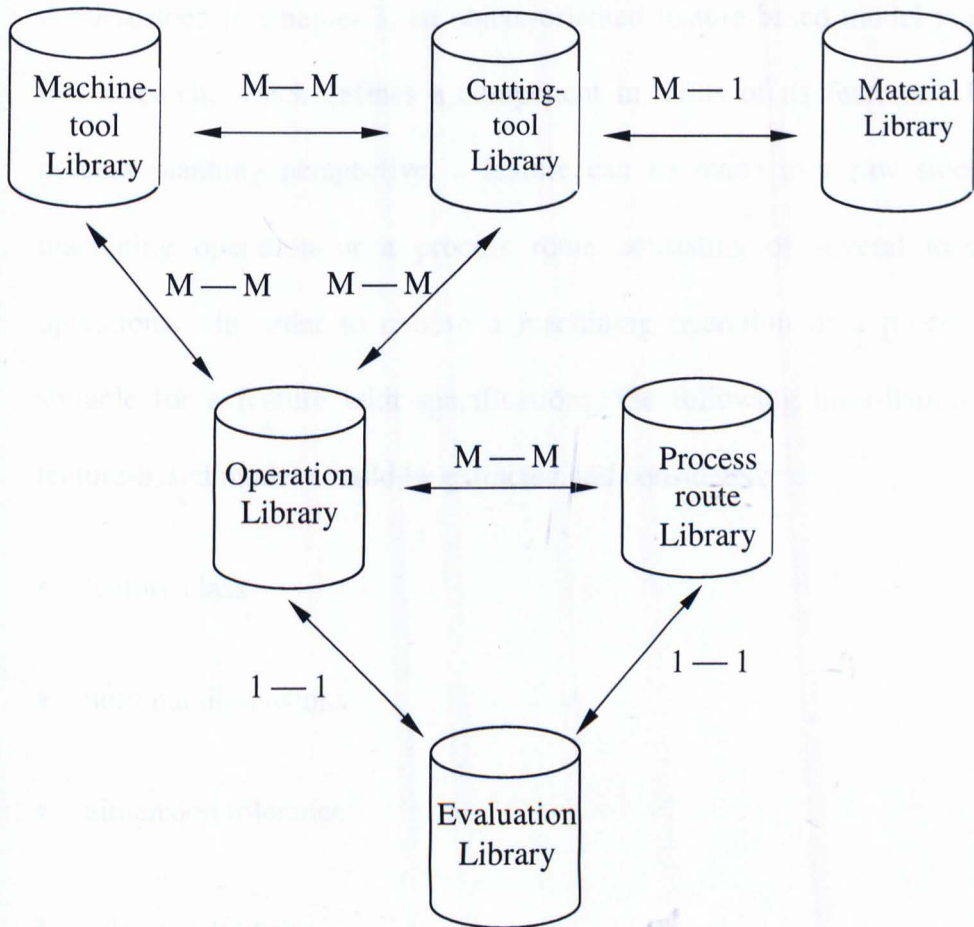
According to various technological standards, manuals and practical industrial environments, many standard process routes have been established for typical features. In the process route library, each feature has one or more standard process routes and a standard process route consists of several machining operations in a prescribed sequence. As a collection of these standard process routes, the process route library gives a frame of the activities for each process route associated with the feature.

6) Evaluation library

Evaluation library concerns information for process sequencing evaluation including exchanging time, exchanging cost and the precedence reward/penalty values between machining operations.

Aiming to increase the effectiveness, it is necessary to build the relationships among the libraries. These relationships are multiple but limited. For example, the relationship of the machine tool library and the cutting tool library is multiple,

where a machine tool can be associated with a set of cutting tools, and a cutting tool can also be used on several machine tools. On the other hand, the number of possible combinations between machine tools and cutting tools is limited, only a specific set of cutting tools can be assembled on a given machine tool. The detailed relations among these libraries are shown in Figure 6.2.



M — M: Many-to-Many relationship. One record in first library can be related to many records in second library, and a record in second library can have many related records in first library.

M — 1: Many-to-One relationship. One record in second library can be related to many records in first library.

1 — 1: One-to-One relationship. The two libraries can have only one record in each that are related to each other.

Figure 6.2 The relationships between libraries

6.4. Selection of machining operations

This sub-module is designed for selecting machining operations with three functions.

1) Extraction of feature information from the feature-based model

As described in Chapter 3, an object-oriented feature-based model is used in this research, which defines a component in terms of its features. From a process planning perspective, a feature can be made in a raw stock by a machining operation or a process route consisting of several machining operations. In order to choose a machining operation or a process route suitable for a feature with specifications, the following information in the feature-based model should be extracted and considered:

- feature class
- nominal dimensions
- dimension tolerance
- surface roughness

2) Interactive input

An interactive interface is designed to input some technological data that the CAD does not provide but the CAPP requires, such as the shape and size of raw material, production batch size, cost and time requirements.

3) Selection of appropriate process routes

Based on the process route library, an appropriate process route can be determined according to design specifications. Because the process route library has defined related facilities for each machining operation, once the operation route has been decided, the required machine tools and cutting tools are automatically selected. The task is described in Figure D.7 (in Appendix D), which can be divided into three steps:

Step 1: Find all feasible process routes from the process route library that are suitable for the given feature class, which can be defined as:

$$PR_i = \{pr_{i1}, pr_{i2}, \dots, pr_{ij}, \dots, pr_{in}\} \quad (6-1)$$

where PR_i is a set of all feasible process routes that can be chosen to machine the i th feature based on its class, and

pr_{ij} is the j th process route candidate to machine the i th feature

$$pr_{ij} = \{O_{ij1}, O_{ij2}, \dots, O_{ijs}, \dots, O_{ijp}\} \quad (6-2)$$

O_{ijs} is the s th machining operation in the j th process route candidate to machine the i th feature, and

$$O_{ijs} = \{P, M, T, TAD, C, MT\} \quad (6-3)$$

where P is operation type,

M is machine tool for the operation,

T is cutting tool for the operation,

TAD is machining approach directions for the operation,

C is operation cost, and

MT is machining time of the operation.

Step 2: Select all feasible process routes that can achieve the requirements (dimensions, tolerances and surface finish) of the given feature from the set PR_i . Based on the comparison between design specifications and machining capability available, all those process routes that cannot satisfy the design specifications are rejected from the list of feasible process routes, that is

$$FPR_i = \{pr_{i1}, pr_{i2}, \dots, pr_{ij}, \dots, pr_{im}\} \quad (6-4)$$

where FPR_i is a set of all feasible process routes to machine the i th feature, which can satisfy the design specifications, and

FPR_i is a subset of PR_i , that is,

$$FPR_i \subset PR_i \quad (6-5)$$

$$pr_{i1} \in PR_i \quad (6-6)$$

$$m \leq n \quad (6-7)$$

Step 3: Evaluate remaining feasible process routes

If a feature has more than three remaining feasible process routes, all remaining feasible process routes for this feature will be evaluated based on their manufacturing costs and three feasible process routes with the minimum manufacturing cost are finally chosen.

6.5. Identification of feature precedence list

Tool Approach Direction (TAD) is defined as a direction from which a cutting tool can access a machinable volume. Here, a valid TAD for a feature should satisfy the following conditions:

- 1) Accessibility: Accessibility means that along the TAD, the cutting tool can be positioned to machine the feature without any interference. If not, the TAD is considered invalid, e.g. interference with other features. The accessibility of TAD can be examined based on the feature relationships (e.g. parent feature and child feature) and its location relative to the adjacent features.
- 2) Tolerance and surface finish requirements: The machining operation along the TAD should not violate tolerance and surface finish requirements of the machining tools.
- 3) Availability of cutting tool: A TAD can be considered valid only if there is a cutting tool available to machine the feature along the TAD.

For a feature, potential TADs are determined at the design stage, but the validity of a TAD depends on the feature class and feature relationships. Here, a feature precedence list specifying the order of all features of a component is built by considering the feature relationships and design intention. The constraints of feature precedence keep all TADs valid for the following stages of process planning.

Based on the definition specified in Chapter 4, the feature relationships can be used as a basis for the generation of a feature-precedence graph, which can be generally defined as

$$G=(P, D) \quad (6-8)$$

$$P = \bigcup_{i=1}^m P_i \quad (6-9)$$

where P is a set of nodes representing the features,

D is a set of directed arcs, each of which represents a precedence relation between two features, and

P_i represents a precedence subset for the features

Some features may belong to two or more precedence subsets, which implies that the subsets $P_1, P_2, \dots, P_i, \dots, P_m$ are not always mutually exclusive. Also, each precedence subset is independent of the precedence relations, which means no directed arcs exist among precedence subsets. For example, the feature-precedence graph for the given component is shown in Figure 6.3. There are four precedence subsets, which can be represented as $P_1 = \{1, 5\}$, $P_2 = \{1, 3, 7\}$, $P_3 = \{1, 3, 4, 2\}$ and $P_4 = \{6, 8, 9\}$. It can be seen that feature 1 and feature 3 exist in three and two subsets, respectively. The non-exclusiveness of precedence subsets may lead to some difficulties for operation sequencing. In this research, to eliminate such non-exclusiveness and reflect the design intention, a feature precedence algorithm has been proposed as the following:

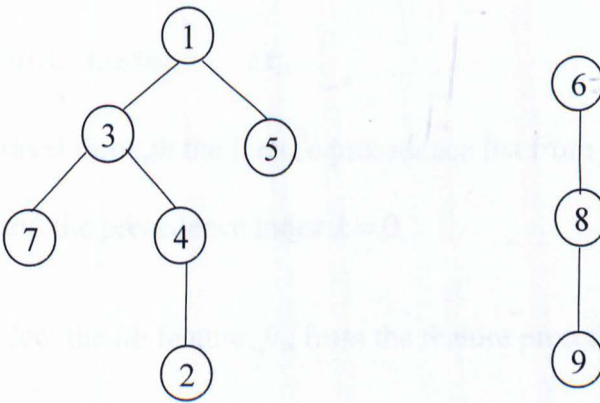
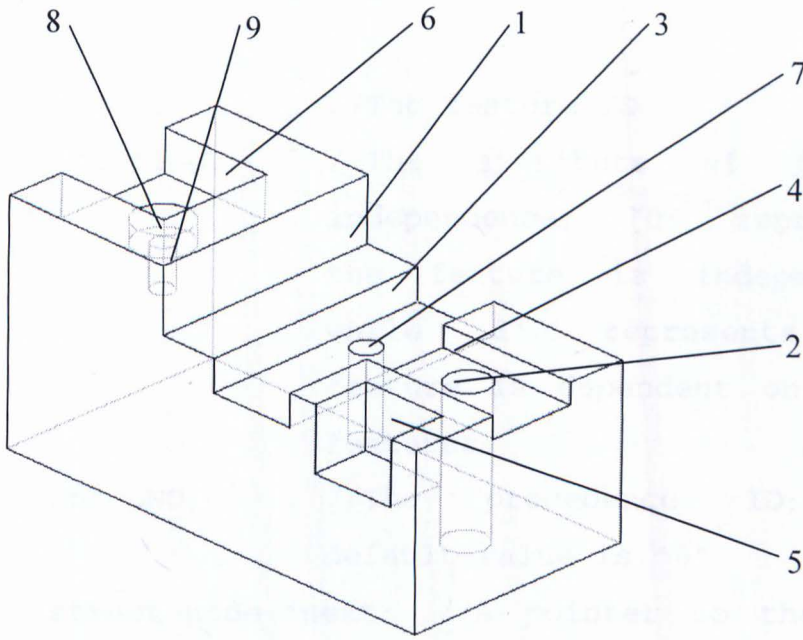


Figure 6.3 An example of feature-precedence graph

Step 0: Initialise feature precedence list, L , based on the design procedure,

$$L = ft_1 \rightarrow ft_2, \dots, \rightarrow ft_n$$

```

struct node
{
    int no;           //The feature ID
    int flag;        //The attribute of feature
                    //independence; '0' represents
                    //the feature is independent,
                    //while '1' represents the
                    //feature is dependent on other
                    //features.
    int pNO;         //The precedence ID; the
                    //default value is "0"
    struct node *next; //A pointer to the next
                    //feature
}
struct node fti

```

Step 1: Travel through the feature precedence list from ft_1 to ft_n , and set iteration index $i = 1$, and the precedence index $k = 0$

Step 2: Select the i th feature, ft_i , from the feature precedence list.

Step 2.1: If ft_i is an independent feature, then go to the next feature ft_{i+1} .

Step 2.2: If ft_i is a dependent feature, then

- 1) If $ft_i.pNO = 0$, then $k = k + 1$, and $ft_i.pNO = k$
- 2) Travel through the feature precedence list from ft_{i+1} to ft_n .

If feature ft_j has a precedence relation with feature ft_i , ($j > i$), then

- If $ft_j.pNO = 0$ then $ft_j.pNO = ft_i.pNO$

- If $ft_j.pNO \neq 0$ and $ft_i.pNO > ft_j.pNO$, then

$$ft_s.pNO = ft_j.pNO, \quad \text{where } ft_s.pNO = ft_i.pNO, \text{ and } s = 1, \dots, n$$

$$ft_s.pNO = ft_s.pNO - 1, \quad \text{where } ft_s.pNO > ft_i.pNO, \text{ and } s = 1, \dots, n$$

- If $ft_i.pNO < ft_j.pNO$, then

$$ft_s.pNO = ft_i.pNO, \quad \text{where } ft_s.pNO = ft_j.pNO, \text{ and } s = 1, \dots, n$$

$$ft_s.pNO = ft_s.pNO - 1, \quad \text{where } ft_s.pNO > ft_j.pNO, \text{ and } s = 1, \dots, n$$

If feature ft_j needs to be machined prior to feature ft_i , ($j > i$), then exchange the positions of feature ft_i and feature ft_j , and set $i = i - 1$

Step 2.3: Set $i = i + 1$. If $i > n$, then go to end. If not, go to Step 2.1.

Based on the above procedure, a complete feature precedence list is built to reflect the design procedure and precedence constraints. Figure D.8 (in Appendix D) summarises the detailed procedure and Table 6.1 provides an example based on the component shown in Figure 6.3.

6.6. Grouping of machining operations

A crucial step in the proposed process planning is to group machining operations. To achieve the objective of minimising changes of setups, machine tools and cutting tools, features that can be machined in the same orientation of the component on the machine table, should be machined together or successively. The feature TADs are closely related to the component orientation on the machine

table. Therefore, operations for features, which have the same TAD and the same precedence constraints, are collected into a group.

Table 6.1 An example of building feature precedence list

Step no		Feature precedence list								
0	list	1	2	3	4	5	6	7	8	9
	pNo									
1	list	1	2	3	4	5	6	7	8	9
	pNo	1		1		1				
2	list	1	2	3	4	5	6	7	8	9
	pNo	1	2	1	2	1				
3	list	1	4	3	2	5	6	7	8	9
	pNo	1	2	1	2	1				
4	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1		1		
5	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1		1		
6	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1		1		
7	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1		1		
8	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1	2	1	2	
9	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1	2	1	2	
10	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1	2	1	2	2
11	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1	2	1	2	2
Final result	list	1	3	4	2	5	6	7	8	9
	pNo	1	1	1	1	1	2	1	2	2

Note:

i : The i th feature is being checked.

i j : The i th and the j th features need to be exchanged.

6.6.1 Determination of TADs

The determination of TADs is a complicated problem concerning with fixture clamping, surface analysis and tolerance. It has been addressed by some

researchers. For example, Kim and Jeong [1996] proposed an algorithm to find feasible TADs for sculptured surface manufacture; Sarma and Wright [1996] presented a method for selecting the access faces to features based on minimising the number of setups, and the number of tool changes; and Yang *et al* [1999] gave a general algorithm to obtain a feasible TAD for sculptured surface machining based on convex analysis. In this research, TAD determination is not a main issue, and therefore, it is simplified based on features and setup changes.

As described in Chapter 3, for prismatic components, TADs are usually specified in the world co-ordinates. Thus, six possible TADs are assumed, i.e. the six normal directions of a prismatic block (+x, -x, +y, -y, +z, -z). In order to determine feature groups with the same TAD, a six-binary ordered vector is defined to represent feature TADs. For example, if a through hole has two candidate TADs, i.e. +y and -y, then its vector is (0, 0, 1, 1, 0, 0). If a through slot has vector of (1, 0, 0, 0, 1, 1), the feature has three candidate TADs, +x, +z and -z. Features with multiple candidate TADs may be collected into more than one group. In order to simplify the situation, features are first evaluated to select their individual tool approach direction. The proposed approach to TAD selection is based on minimum setup change, which is shown in Figure D.9 (in Appendix D).

Step 0: Let

UTAD be a set of features whose TADs have not been determined yet.

$$UTAD = \{f_1, \dots, f_i, \dots, f_n\}$$

where n is the number of the features whose TAD has not been determined yet.

FT_i be a set of TAD candidates that the i th feature, $f_i \in UTAD$, has for its three candidate process routes.

$$FT_i = \{TAD_{i1}, \dots, TAD_{ik}\}, i = 1, 2, \dots, n$$

where $0 < k \leq 3$

Step 1: Calculate the number of features in $UTAD$, NT_j , which have the corresponding candidate TADs, j ($j=1, 2, \dots, 6$) represent six candidate TADs: $+x$, $-x$, $+y$, $-y$, $+z$ and $-z$.

Step 2: Choose the TAD_j , $D=TAD_j$, where

$$NT_j \geq NT_{k \neq j}$$

Step 3: Determine the TAD for all features, f_i , which have a TAD candidate= D , that is,

$$D \in FT_i$$

Then, delete these features from $UTAD$.

Step 4: If $UTAD = \emptyset$, then go to end. Else, go to Step 1.

Step 5: End.

6.6.2 Grouping algorithm

The machining operations for the features with the same TAD will be grouped.

The detailed steps are described below.

Step 1: Initialise the first group, G_m , $m=1$, by inserting the first feature f_i from the feature precedence list, L , into G_m .

Step 2: Select a new feature, f_i , from L . If L is NULL, then go to end.

Step 3: Compare the TAD of the new feature f_i with the TADs of the candidate groups. If there exists a group with the same TAD, G_j , then go to next step. If not, go to the step 5.

Step 4: If the feature has the same precedence constraints with the features in group G_j , then insert feature f_i into G_j . Delete feature f_i from L , and go to step 2. If not, then go to next step.

Step 5: Let $m=m+1$. Create a new group G_m and insert feature f_i into G_m . Delete feature f_i from L . Add G_m into the candidate groups. Go to Step 2.

The above procedure is summarised in Figure 6.4.

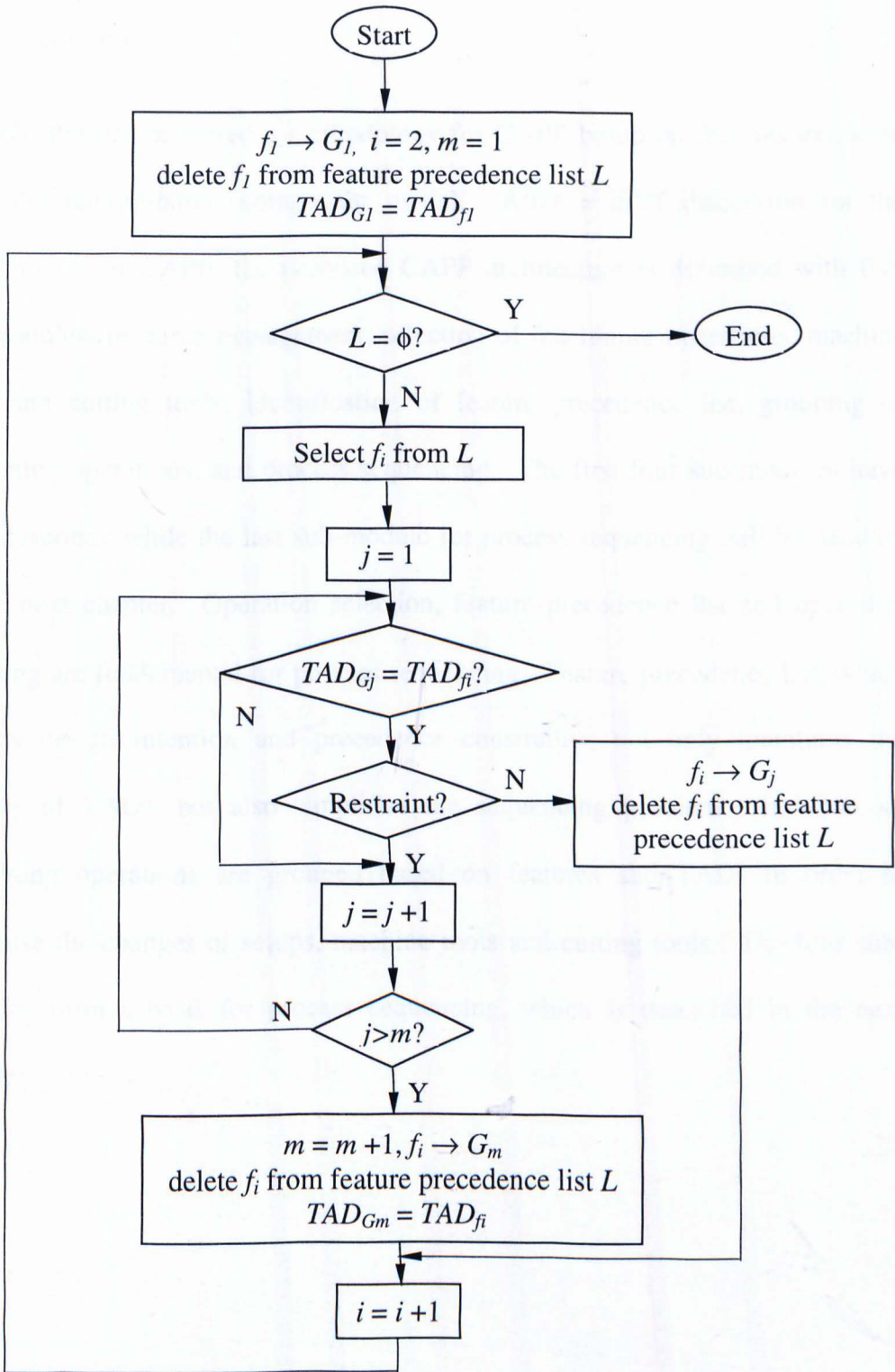


Figure 6.4 Procedure of grouping machining operations

6.7. Summary

This chapter has proposed a methodology for CAPP based on features extracted from the feature-based component model. After a brief discussion on the requirements of CAPP, the proposed CAPP architecture is described with five sub-modules: resource management, selection of machining operations, machine tools and cutting tools, identification of feature precedence list, grouping of machining operations, and process sequencing. The first four sub-modules have been described while the last sub-module for process sequencing will be detailed in the next chapter. Operation selection, feature precedence list and operation grouping are fundamental for process sequencing. Feature precedence list, which reflects design intention and precedence constraints, not only maintains the validity of TADs, but also simplifies the sequencing problem. In addition, machining operations are grouped based on features and TADs in order to minimise the changes of setups, machine tools and cutting tools. The four sub-modules form a basis for process sequencing, which is described in the next Chapter.

Chapter 7.

Process Sequencing

In developing computer-aided process planning (CAPP) systems, the determination of the operation sequence is one of the most important tasks and also a bottleneck task in the process [Qiao *et al*, 2000]. As a complex decision-making process, process sequencing is influenced by several constraints, such as tool accessibility, tolerance requirement, feature relationships, cost and time, etc. Therefore, manufacturing feasibility, production economy and optimal utilisation of manufacturing resources need to be considered and some artificial intelligence (AI) reasoning techniques are required.

This chapter determines the process sequence required to produce the component with the objective of optimising machining cost and time, while satisfying the precedence constraints. As described in Chapter 6, process sequencing is based on the operation selection made at a previous stage. The process routes for features, and machine and tool information are determined in advance and hence no process alternatives are considered at the stage of sequencing. It is also assumed that the cost and time of machining operations, and the change cost and time between any two operations are given in advance and deterministic. In addition, the problem of grouping machining operations has been done (see Chapter 6) and the sequence is based on these machining operation groups.

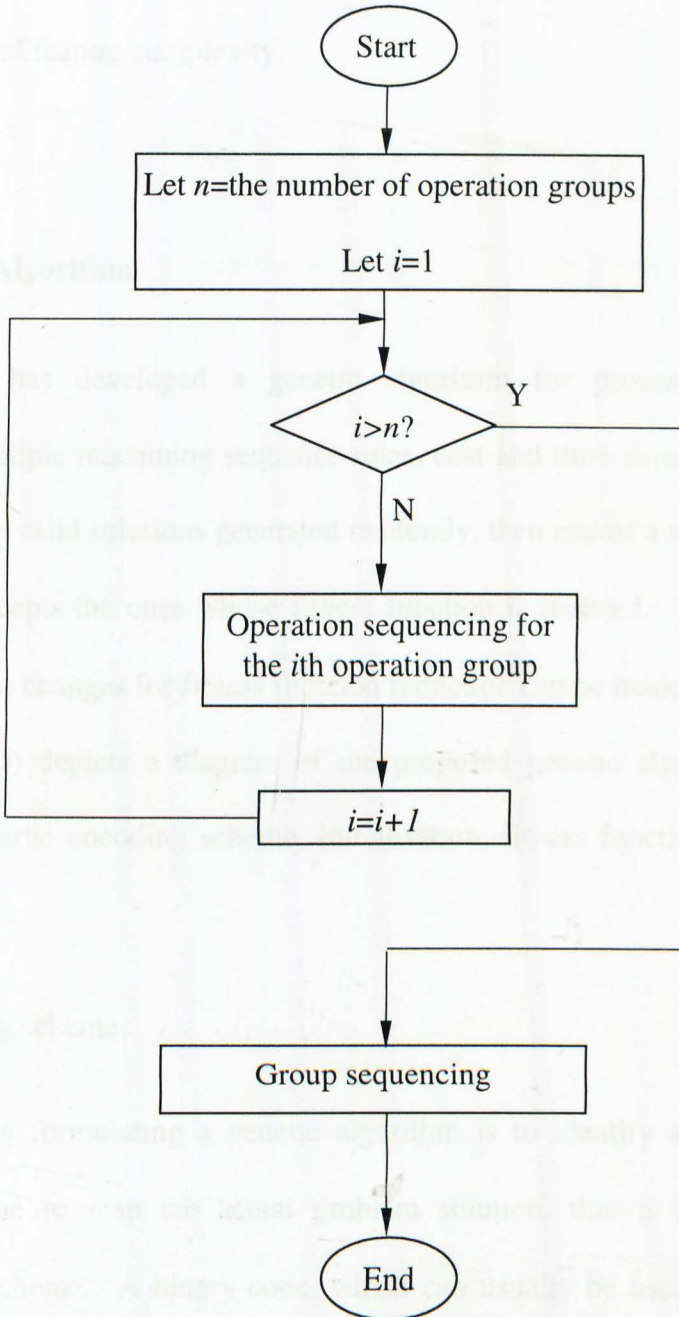


Figure 7.1 The procedure for process sequencing

As shown in Figure 7.1, the proposed process sequencing includes two stages: operation sequencing within a group and sequencing of operation groups. This chapter discusses process sequencing covering operation sequencing in a group and sequencing of operation groups using a genetic algorithm, evaluation of

sequencing rules, evaluation of cost and time, determination of relative weights and evaluation of feature complexity.

7.1 Genetic Algorithm

This research has developed a genetic algorithm for process sequencing, considering multiple machining sequence rules, cost and time simultaneously. It starts with some valid solutions generated randomly, then makes a random change to them and accepts the ones whose fitness function is reduced. The process is repeated until no changes for fitness function reduction can be made. Figure D.10 (in Appendix D) depicts a diagram of the proposed genetic algorithm, which contains four parts: encoding scheme, initialisation, fitness function calculation and operations.

7.1.1 Encoding scheme

The first step in formulating a genetic algorithm is to identify an appropriate encoding scheme to map the actual problem solution, that is genetic string representation scheme. A binary code, which can usually be used in a genetic algorithm, is not suitable for direct use in operations sequencing, especially for complex components, as longer gene code chains will result. In this case, based on the precedence constraints, a numeric code consisting of non-negative numbers is devised to solve problem of sequencing, called chromosome.

1) Chromosomes for machining operations sequencing

Supposing there are m_i machining operations in a group (e.g. G_i), the operations can be denoted as follows:

$$G_i = \{O_{i1}, O_{i2}, \dots, O_{ij}, \dots, O_{im_i}\}$$

where O_{ij} is the j th machining operation in the i th machining operation group

Any sequence of all machining operations in the group is a possible solution for operation sequence of the group. Therefore, a chromosome for operation sequence is defined, which consists of m_i bits, and each bit represents an operation once and only once. A bit (operation) in the chromosome can be represented as:

```
struct pChromosome_Bit
{
int O_no;           //The ID of the operation
int feature_no     //The ID of the feature that the
                   //operation is used for
int flag;          //The sign of feature
                   //relationship; '0' represents that
                   //the feature has no relationships
                   //with other features, and '1'
                   //represents that the feature has
                   //some precedence relationships
                   //with other features.

int M_no;          //The ID of the machine tool to
                   //execute the operation
int T_no;          //The ID of the cutting tool to
                   //execute the operation
```



```

direction D_no;          //The ID of the TAD for the
                        operation
double pCost;           //The cost caused by the operation
double pTime;          //The time needed for the operation
}

pChromosome_Bit pOperation[]=new pChromosome_Bit[mi];

Struct pChromosome
{
int num;                //The number of operation groups
pChromosome_Bit pOperation[]=new pChromosome_Bit[mi];
double pv_fitness;     //The value of fitness function
}

```

The number of bits in the operation chromosome (i.e. m_i) is equal to the total number of operations in the group. The sequence of the bits describes the machining operation sequence of the group. Thus, the total number of possible operation sequences for the group is $m_i!$.

2) Chromosomes for operation group sequencing

Supposing there are n operation groups for machining a component, the operation groups can be denoted as follows:

$$P_G = \{G_1, G_2, \dots, G_b, \dots, G_n\}$$

where P_G is a set of machining operation groups, and

G_i is the i th machining operation group.

The sequence for machining operations in each group is generated first, and thus, any sequence of all the operation groups in P_G is a possible solution of process sequence. Similarly, a chromosome for group sequence is defined, which consists of n bits, and each bit represents an operation group once and only once. The n is equal to the total number of operation groups. The chromosome can be described as:

```

struct gChromosome_Bit
{
int G_no;           //The ID of the operation group
int flag;          //The sign of the group's state;
                  // '0' represents that the operation
                  // group is independent, and '1'
                  // represents the operation group has
                  // some precedence relationships with
                  // other groups.
int flagno;        //The ID of precedence constraints
                  // the operation group belongs to.
double gCost;      //The cost caused by the group
double gTime       //The time needed for the group
pChromosome_Bit pOperation[] //Operations sequence
                              in the group
}

gChromosome_Bit pGroup[]=new Chromosome_Bit[n];

Struct gChromosome
{
int num;           //The number of operation groups
Chromosome_Bit pGroup[]=new Chromosome_Bit[n];
double gv_fitness; //The value of fitness function
}

```

Based on the above definitions, the group chromosomes and operation chromosomes describe the process sequence. All the combinations of all group chromosomes and all operation chromosomes constitute the possible solution space. Thus, if there are no precedence requirements, the total number of possible process sequences is

$$n! \prod_{i=1}^n m_i!$$

7.1.2 Initial populations

An initial population of solutions consists of a population of randomly generated solutions to the problem at hand [Yip-Hoi and Dutta, 1996]. Initialisation is a process that generates several initial populations for a component with n operation groups. The initial populations generated should be spread sufficiently over the search space to represent as wide a variety of solutions as possible. In the meantime, when an initial solution is created, the precedence constraints should be considered. This is because certain operation sequences gained randomly may be infeasible with respect to the precedence relationships. In order to eliminate such infeasible sequences while obtaining the initial population, this research has devised two initial precedence constraint algorithms for operation sequencing and group sequencing, respectively. A valid sequence solution is defined as the one that satisfies all the precedence constraints, such as geometrical precedence and manufacturing precedence. The total number of strings in the initial population is set to '18'.

1) Initial precedence constraint algorithm for operation sequencing

As described in Chapter 6, an operation group consists of all operations for the features, which have the same TAD and the same precedence constraints. Thus, only one precedence constraint needs to be considered here, that is, the order of operations required for each feature.

Assuming a group has m features, and each feature (feature i) has p_i machining operations, then the number of machining operations in the group is

$$pnum = \sum_{i=1}^m p_i .$$

Step 0: Set $i=1$.

Step 1: Choose p_i positions in the initial chromosome that have not been occupied randomly.

Step 2: Put these p_i machining operations for the i th feature into the chosen positions in the required order.

Step 3: Let $i=i+1$, and $pnum=pnum- p_i$.

Step 4: If $i \leq m$, then go to Step 1, else go to end.

Step 5: End.

2) Initial precedence constraint algorithm for group sequencing

The proposed initial precedence constraint algorithm is based on the precedence relationships determined in Chapter 6. Here, it is illustrated by an example. Assuming a process including n operation groups (i.e. $n=12$), that is

$G1, G2, \dots, G12$. Thus, the initial chromosome has n bits. There are K constraint precedence relationships (i.e. $K=2$) as following:

- Precedence relationship I ($k=1$): $G6 \rightarrow G5 \rightarrow G1 \rightarrow G9$
- Precedence relationship II ($k=2$): $G12 \rightarrow G4 \rightarrow G11$

For other groups ($G2, G3, G7, G8, G10$), there are no constraint precedence relationships ($k=0$).

Firstly, appropriate positions are obtained for the groups in precedence relationship I ($k=1$): $G6 \rightarrow G5 \rightarrow G1 \rightarrow G9$

- $G6$: According to precedence relationship I, there must be three unused positions after the position of $G6$. In other words, although twelve positions ($u=12$) have not been used, only nine positions that $G6$ can choose, $m \in [1,9]$, e.g. 3, that is, position 3 in the chromosome.
- $G5$: After the position of $G6$, there are only nine unused positions, while $G5$ has to leave two unused positions for $G1$ and $G9$. Thus, $m \in [1,7]$, e.g. 4, that is, position 7 in the chromosome.
- $G1$: After the position of $G5$, there are only five unused positions, while $G1$ has to leave one unused position for $G9$. Thus, $m \in [1,4]$, e.g. 1, that is, position 8 in the chromosome.
- According to the above process, it can be gained that $m \in [1,4]$, e.g. 2 for $G9$, that is, position 10 in the chromosome.

Then, for the precedence relationship II ($k=2$) groups, $G12 \rightarrow G4 \rightarrow G11$, the same process is applied.

- $G12$: After the arrangement for the precedence relationship I groups, there are still eight positions unused ($u=8$). In order to leave two unused positions for $G4$ and $G11$, $m \in [1, 6]$, e.g. 2, that is, position 2 in the chromosome.
- $G4$: Behind the position of $G12$, there are six positions while five positions can be used as candidates for $G4$, $m \in [1, 5]$, e.g. 4, that is, position 9 in the chromosome.
- $G11$: likewise, $m \in [1, 2]$, e.g. 1, that is, position 11 in the chromosome.

Finally, appropriate positions are determined for the other groups ($G2, G3, G7, G8, G10$), for which $k=0$. Because there are no precedence constraints between these groups, the identification for random numbers is straightforward.

- $G2$: There are five candidate positions so that the random number m can be identified in the field of $[1, 5]$, e.g. 3, that is, position 5 in the chromosome.
- $G3$: Only 4 candidate positions are available for $G3$. The random number $m \in [1, 4]$, e.g. 1, that is, position 1 in the chromosome.

Similarly, the positions of $G7$, $G8$, $G10$ can be determined in the fields of [1,3], [1,2] and [1,1], respectively. Here, $m=3$ for $G7$ (i.e. position 12), $m=1$ for $G8$ (i.e. position 4), and $m=1$ for $G10$ (i.e. position 6).

The detailed process and change of parameters are displayed in Table 7.1 and Figure D.11 (in Appendix D). Applying the initial precedence constraint algorithm, the initial population is ensured in the feasible domain.

7.1.3 Fitness function

In the application of a genetic algorithm to process sequencing, the fitness function is a performance criterion, which indicates the degree of objective satisfaction of a searched solution. After a certain number of searches, if the value of fitness function does not decrease, it can be identified that the fitness function has reached the optimal point (the least total value) and the searched operation sequence is satisfactory with the goal. Then, the search process stops. Fitness calculation is considered as an essential ingredient and the most critical step for a GA method. Several fitness functions are provided for process sequencing, which include minimum number of setups, minimum machining cost and shortest processing time, etc. However, previous research has only considered individual criteria and therefore the outcome solution is not overall optimal. This research has developed an integrated optimisation strategy to obtain an overall optimal process sequence.

Table 7.1. Example of Initial precedence constraint algorithm

$n=12, K=2$																																										
	m	$[x_0, x_1]$	j	u																																						
G6	3	[1,9]	3	12	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="12" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G6</td><td colspan="9"></td></tr> </table>																G6																					
			G6																																							
G5	4	[1,7]	7	11	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G6</td><td colspan="9" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G6</td><td colspan="3"></td><td>G5</td><td colspan="6"></td></tr> </table>				G6													G6				G5																
			G6																																							
			G6				G5																																			
G1	1	[1,4]	8	10	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td colspan="6" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G6</td><td colspan="3"></td><td>G5</td><td>G1</td><td colspan="6"></td></tr> </table>				G6				G5										G6				G5	G1														
			G6				G5																																			
			G6				G5	G1																																		
G9	2	[1,4]	10	9	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td colspan="6" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G6</td><td colspan="3"></td><td>G5</td><td>G1</td><td colspan="2"></td><td>G9</td><td colspan="3"></td></tr> </table>				G6				G5	G1										G6				G5	G1			G9										
			G6				G5	G1																																		
			G6				G5	G1			G9																															
G12	2	[1,6]	2	8	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td style="background-color: #cccccc;"></td><td>G9</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G12</td><td>G6</td><td colspan="3"></td><td>G5</td><td>G1</td><td colspan="2"></td><td>G9</td><td colspan="3"></td></tr> </table>				G6				G5	G1		G9							G12	G6				G5	G1			G9										
			G6				G5	G1		G9																																
			G12	G6				G5	G1			G9																														
G4	4	[1,5]	9	7	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G12</td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td style="background-color: #cccccc;"></td><td>G9</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G12</td><td>G6</td><td colspan="3"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td colspan="3"></td></tr> </table>				G12	G6				G5	G1		G9							G12	G6				G5	G1	G4	G9										
			G12	G6				G5	G1		G9																															
			G12	G6				G5	G1	G4	G9																															
G11	1	[1,2]	11	6	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G12</td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G12</td><td>G6</td><td colspan="3"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="3"></td></tr> </table>				G12	G6				G5	G1	G4	G9							G12	G6				G5	G1	G4	G9	G11									
			G12	G6				G5	G1	G4	G9																															
			G12	G6				G5	G1	G4	G9	G11																														
G2	3	[1,5]	5	5	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G12</td><td>G6</td><td colspan="3" style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G12</td><td>G6</td><td colspan="2"></td><td>G2</td><td colspan="2"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="2"></td></tr> </table>				G12	G6				G5	G1	G4	G9	G11							G12	G6			G2			G5	G1	G4	G9	G11						
			G12	G6				G5	G1	G4	G9	G11																														
			G12	G6			G2			G5	G1	G4	G9	G11																												
G3	1	[1,4]	1	4	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3" style="background-color: #cccccc;"></td><td>G12</td><td>G6</td><td style="background-color: #cccccc;"></td><td>G2</td><td style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td colspan="2"></td><td>G2</td><td colspan="2"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="2"></td></tr> </table>				G12	G6		G2		G5	G1	G4	G9	G11							G3	G12	G6			G2			G5	G1	G4	G9	G11					
			G12	G6		G2		G5	G1	G4	G9	G11																														
			G3	G12	G6			G2			G5	G1	G4	G9	G11																											
G7	3	[1,3]	12	3	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td style="background-color: #cccccc;"></td><td>G2</td><td style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td colspan="3" style="background-color: #cccccc;"></td></tr> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td colspan="2"></td><td>G2</td><td colspan="2"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td>G7</td><td colspan="2"></td></tr> </table>				G3	G12	G6		G2		G5	G1	G4	G9	G11							G3	G12	G6			G2			G5	G1	G4	G9	G11	G7			
			G3	G12	G6		G2		G5	G1	G4	G9	G11																													
			G3	G12	G6			G2			G5	G1	G4	G9	G11	G7																										
G8	1	[1,2]	4	2	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td style="background-color: #cccccc;"></td><td>G2</td><td style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td>G7</td><td colspan="3"></td></tr> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td>G8</td><td>G2</td><td colspan="2"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td>G7</td><td colspan="3"></td></tr> </table>				G3	G12	G6		G2		G5	G1	G4	G9	G11	G7							G3	G12	G6	G8	G2			G5	G1	G4	G9	G11	G7			
			G3	G12	G6		G2		G5	G1	G4	G9	G11	G7																												
			G3	G12	G6	G8	G2			G5	G1	G4	G9	G11	G7																											
G10	1	[1,1]	6	1	<table border="1" style="width: 100%; text-align: center;"> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td>G8</td><td>G2</td><td style="background-color: #cccccc;"></td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td>G7</td><td colspan="3"></td></tr> <tr><td colspan="3"></td><td>G3</td><td>G12</td><td>G6</td><td>G8</td><td>G2</td><td>G10</td><td>G5</td><td>G1</td><td>G4</td><td>G9</td><td>G11</td><td>G7</td><td colspan="3"></td></tr> </table>				G3	G12	G6	G8	G2		G5	G1	G4	G9	G11	G7							G3	G12	G6	G8	G2	G10	G5	G1	G4	G9	G11	G7				
			G3	G12	G6	G8	G2		G5	G1	G4	G9	G11	G7																												
			G3	G12	G6	G8	G2	G10	G5	G1	G4	G9	G11	G7																												

1) Fitness function for operation sequencing in a group

This is executed prior to group sequencing, and therefore, its fitness function is described as the degree that the sequence satisfies process sequence rules. An evaluation is proposed based on an evaluating indicator hierarchy and analytical hierarchy process method. The details are presented in Section 7.2.

2) Fitness function for group sequencing

Group sequencing is the last stage of process sequencing. The fitness function for group sequencing should be the final objective of process sequencing, that is, to find an optimal point satisfying process sequence rules, minimum machining cost and shortest processing time simultaneously. Thus, the fitness function is defined by the following expression:

$$F = w_m f_m + w_c f_c + w_t f_t \quad (7-1)$$

where F is the fitness,

f_m is the degree of satisfaction with process sequence rules,

f_c is the relative evaluating value for manufacturing cost,

f_t is the relative evaluating value for manufacturing time, and

w_m , w_c , and w_t are the weights for the above evaluations, respectively.

A calculation supporting the fitness function has been developed based on the above expression. It consists of three parts, that is, evaluation for process sequence rules, evaluation for manufacturing cost and time, and collation of

weights. The first two parts are described in Sections 7.2 and 7.3, respectively. In Sections 7.4 and 7.5, a trained neural network and a fuzzy evaluation are presented for the collation of weights.

7.1.4 Operators

The design of appropriate genetic operators including the selection, crossover and mutation plays a major role for the successful genetic algorithm.

1) Selection: Selection is the genetic operator that chooses parents for reproduction in the next generation. The chosen parents will have the chance to be used in the next genetic operation, such as crossover and mutation.

Three selection strategies are usually chosen. They are:

- **Roulette wheel selection:** The chance of a chromosome to be selected is based on their fitness value. The chromosomes are more likely to be selected if they are fitter.
- **Tournament:** First, a small subset of chromosomes is selected at random from the population. After the selection, the one with the best fitness in the tournament is selected to be a parent.
- **Random:** Parents are simply selected completely at random from the population.

In this algorithm, the 'roulette wheel selection' strategy is employed to expedite the search and guarantee the search in an increased trend.

2) Crossover: After the parents are selected, the crossover is applied to the population of the selected parents. Crossover is the genetic operator that creates a new solution in the next generation by splitting and recombining between two parents. There are many types of crossover. Here, three most commonly used crossovers are introduced:

- Single-point crossover: Two parents are split at the same position and recombined with the left of one and the right of the other. For example, if the split point was chosen randomly as 3, two parents used for generating new chromosomes are:

Parent I: G1-G2-G3-G4-G5

Parent II: G4-G2-G1-G5-G3

Then, the new children created are:

Child I: G1-G2-G3I-G5-G3

Child II: G4-G2-G1I-G4-G5

As the classic form of the crossover, single-point crossover is simple but very slow.

- Multi-point crossover: Two parents are split at several randomly chosen sites and recombined into two new children. For instance, if there are two split points 6 and 10, and the parents used for generating new chromosomes are:

Parent I: G1-G2-G3-G4-G5-G6-G7-G8-G9-G10-G11-G12-G13

Parent II: G6-G2-G3-G7-G1-G5-G10-G4-G13-G11-G12-G8-G9

Then, the new children created are:

Child I: G1-G2-G3-G4-G5-G6-G10-G4-G13-G11-G11-G12-G13

Child II: G6-G2-G3-G7-G1-G5-G7-G8-G9-G10-G12-G8-G9

Comparing to single-point crossover, multi-point crossover will produce more mixing, although it may be more disrepute.

- Uniform crossover: Each bit on the child is selected randomly from the corresponding bit of the parents. A mask is used to determine which parent contributes its bit value to which child at the same position. An example is provided, where the parents used for generating new chromosomes are:

Parent I: G1-G2-G3-G4-G5-G6-G7-G8

Parent II: G6-G2-G7-G3-G1-G5-G8-G4

Mask: 11010010

Then, the new children created are:

Child I: G1-G2-G7-G4-G1-G5-G7-G4

Child II: G6-G2-G3-G3-G5-G6-G8-G8

By employing above crossover operators, it is clear that such exchanges may result in an invalid sequence because

- Some operations may appear in the child chromosomes more than once;
- Some operations may not appear in the child chromosomes;
- The precedence constraints may be violated.

In order to overcome the above limitation and gain a valid process sequence, a modified crossover operator is proposed by Li *et al* [2002]:

Step 1: Two chromosomes in the populations are randomly chosen as two parents (e.g. Parent I and Parent II).

Step 2: A splitting point is randomly determined, and each parent chromosome is separated as left and right parts from the splitting point, that is, Parent I-Left, Parent I-Right, Parent II-Left, and Parent II-Right.

Step 3: The left part of Parent I (Parent I-Left) is copied as the left part of child I.

Step 4: The bits in the right part of Parent I are copied to the right part of child I according to their sequences in Parent II.

Step 5: Similarly, child II can be obtained with the left part of Parent II and the bits in the right part of Parent II in their sequences in Parent I.

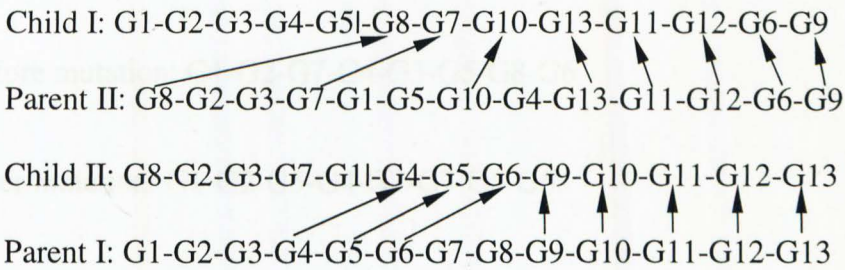
The above precedence is illustrated in the following example:

Supposing there is a split point 5, and the parents used for generating new chromosomes are:

Parent I: G1-G2-G3-G4-G5-G6-G7-G8-G9-G10-G11-G12-G13

Parent II: G8-G2-G3-G7-G1-G5-G10-G4-G13-G11-G12-G6-G9

Then, the new children created are:



Based on the above process, it can be proved that the modified crossover not only guarantees that each operation or operation group appears in a child chromosome once and only once, but also follows the precedence constraint relationships between operations or operation groups in parents with maximum possibilities. Combining with a fitness function considering precedence constraints between operations or operation groups, the precedence constraint adjustment is not required for the modified crossover operator.

- 3) Mutation: After the parents are crossed over, the mutation operator is applied to the population of selected child chromosomes. Mutation is the genetic operator that randomly changes one or more of the chromosomes' gene. It is carried out according to the rate of mutation. The mutation rate should be kept very low (usually about 0.1) as a high mutation rate will destroy fit strings and degenerate the algorithm into a random walk, with all the associated problems.

The purpose of mutation is to provide the populations with new possible solutions that may have been lost during successive generations and prevent the genetic population from converging to a local optima. In the proposed method, one type of mutation strategy is applied: two operation positions are chosen randomly in a chromosome, and then the values in the two positions are exchanged. For example, if the two random operation positions are 3 and 6, and the chromosome to be mutated is:

Before mutation: G1-G2-G7-G4-G3-G5-G8-G6

After mutation: G1-G2-G5-G4-G3-G7-G8-G6

In theory, a constraint adjustment should be applied after the mutation operator. However, considering the fitness function having included precedence constraints, it seems not necessary to adjust the process sequence in a feasible domain using an extra algorithm.

7.1.5 Stop criteria

There are several stop criteria for the search process of a GA. Usually, the fitness function is considered near optimal if its value does not decrease, and the search process stops. The final searched solution is regarded as the result satisfying the goal. According to the result of experiments, this stop criterion is the most suitable for process sequencing.

7.2 Evaluation of manufacturing rules for process sequence

This research uses an analytical hierarchy process (AHP) [Saaty, 1980, 1990, 1994] to evaluate machining sequence rules for process sequencing. As a systematic method for comparing a list of objectives, AHP makes it possible to evaluate objects based on a large number of qualitative and quantitative factors. The proposed AHP model for evaluation is depicted in Figure D.12 (in Appendix D).

Step 1: Identifying all relevant and important process sequence rules as evaluating criteria.

Step 2: Structuring these criteria into hierarchy levels from an overall objective to various criteria and sub-criteria.

Step 3: Determining the relative weights of structured criteria through pairwise comparison, which is built by the judgements of experts.

Step 4: Employing Eigenvalue technique for computing the weights under the AHP. According to the characteristics of the AHP hierarchy, the calculating procedure is in the downward direction along the hierarchy, which means that the weights of higher-level criteria are first computed, and then used by the weight calculation for the lower level criteria.

Step 5: Identifying the evaluating value for the satisfactory degree for process sequence rules.

7.2.1 Process sequence rules

The process sequence rules are derived from both manufacturing requirements, and geometrical and topological information. In order to accommodate the complex rules, an evaluating indicator hierarchy for process sequence rules needs to be defined. The evaluating indicator hierarchy should have: 1) appropriate number and levels of criteria, which provides an overall view of the complex relationship inherent in the manufacturing rules; and 2) an open and adaptive structure, which ensures the possibility to modify and scale the hierarchy with ease to adapt environment changes. The data structure for evaluation criteria is defined in Figure 7.2.

Qiao *et al* [2000] proposed four types of process planning rules, including precedence rules, clustering rules, adjacent order rules and objectives. In the present research, an evaluating criteria hierarchy has been developed based on their work. It consists of two levels, and three main manufacturing sequence constraints are included in the first level.

1) Precedence constraints

- A parent feature should be processed before its child features. Parent and child features have been defined in Chapter 4.
- Rough machining operations should be done before finish machining operations.

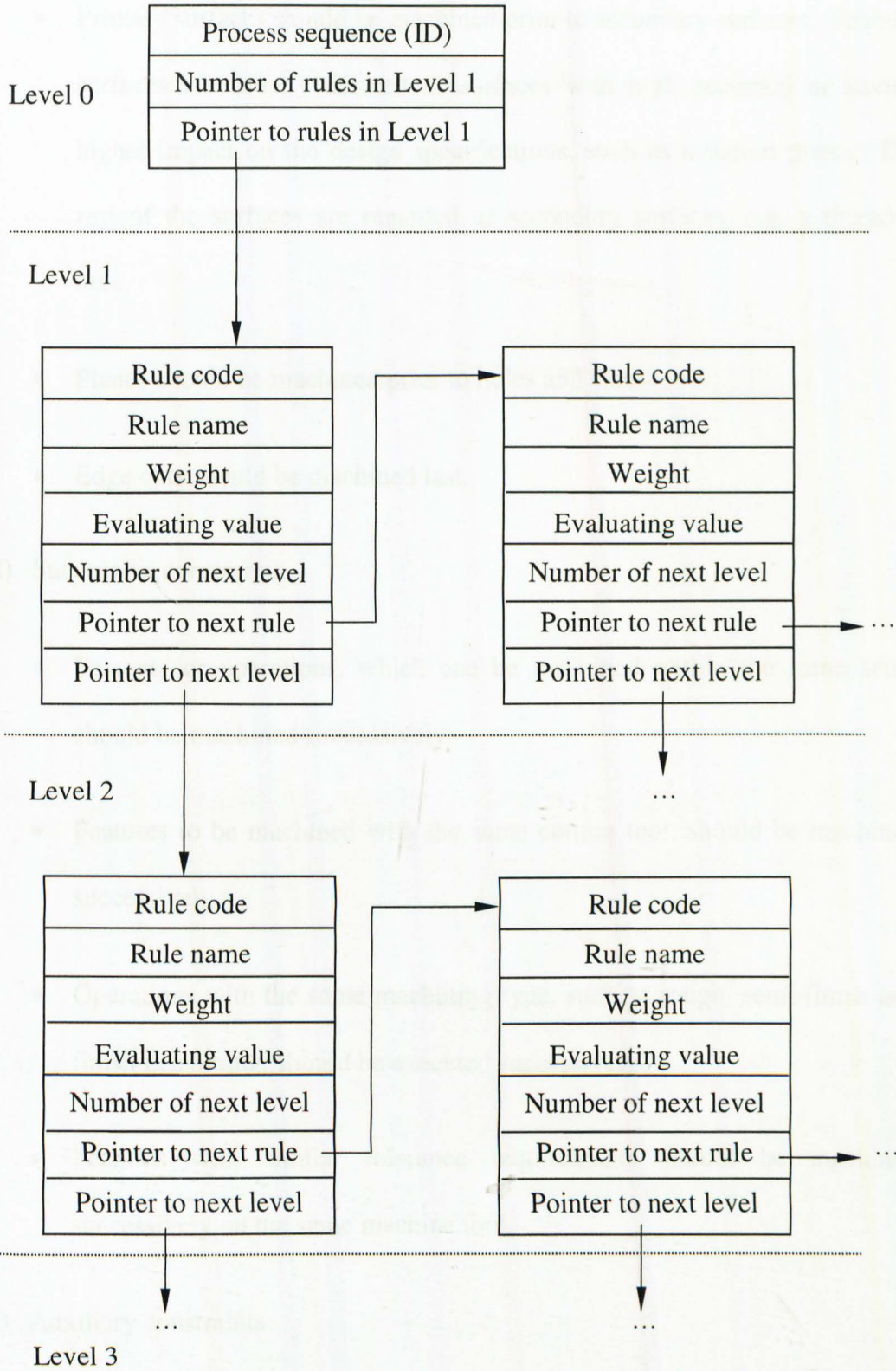


Figure 7.2 Data structure of evaluation criteria

- Primary surfaces should be machined prior to secondary surfaces. Primary surfaces are usually defined as surfaces with high accuracy or having higher impact on the design specifications, such as a datum plane. The rest of the surfaces are regarded as secondary surfaces, e.g. a threaded hole.
- Planes should be machined prior to holes and slots.
- Edge cuts should be machined last.

2) Successive constraints

- Features or operations, which can be machined within the same setup should be machined successively.
- Features to be machined with the same cutting tool, should be machined successively.
- Operations with the same machining type, such as rough, semi-finish and finish machining, should be executed successively.
- Features with similar tolerance requirements should be machined successively on the same machine tool.

3) Auxiliary constraints

- Annealing, normalising and ageing operations of ferrous metal component should be arranged before rough machining or between rough and semi-finish machining.

- Quenching for ferrous metal workpieces should be arranged between semi-finish and finish machining or between rough and semi-finish machining if it is followed by high temperature tempering. Quenching for non-ferrous metals should be arranged between rough and semi-finish machining or before rough machining.
- Carburizing would be arranged between semi-finish and finish machining.

7.2.2 Analytical hierarchy process (AHP)

In practice, it can be impossible to satisfy all sequence rules in a process sequence. For example, a high accuracy hole as the datum surface should be machined with a high priority according to the primary surfaces rule, but it may be in conflict with the rule of planes prior to holes and slots. Therefore, it is necessary to derive a set of numerical weights representing the relative importance of the rules with respect to the manufacturing environment. The importance weights are determined by AHP. Essentially, AHP employs pairwise comparisons of selection criteria so as to enhance objectivity and downplay too much subjectivity [Saaty, 1990]. In the following section, AHP is discussed further with three points: construction of pairwise comparison matrix, calculation of relative weights, and calculation of evaluating value.

1) Construction of pairwise comparison matrix

The pairwise comparison is formed by comparing each rule with all the remaining ones at a certain level. Correspondingly, a pairwise comparison

matrix, called R-matrix, is defined, where the number in the i th row and j th column, r_{ij} , gives the relative importance of the i th process sequence rule as compared with the j th process sequence rule. This can be described as:

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1i} & \cdots & r_{1m} \\ \vdots & & \vdots & & \vdots \\ r_{i1} & \cdots & r_{ii} & \cdots & r_{im} \\ \vdots & & \vdots & & \vdots \\ r_{m1} & \cdots & r_{mi} & \cdots & r_{mm} \end{bmatrix} \quad (7-2)$$

where $i = 1, 2, \dots, m$;

m is the number of selected rules.

$$r_{ii} = 1;$$

$$r_{ij} = 1/r_{ji}$$

Table 7.2. Evaluating criteria for the pairwise comparison matrix

Definition	Intensity of importance(r_{ij})	Intensity of importance(r_{ji})
The i th rule and the j th rule have equal importance	1	1
The i th rule is slightly more important than the j th rule	3	1/3
The i th rule is more important than the j th rule	5	1/5
The i th rule is much more important than the j th rule	7	1/7
The i th rule is absolutely more important than the j th rule	9	1/9
Intermediate values between adjacent scale values	2, 4, 6, 8	1/2, 1/4, 1/6, 1/8

Table 7.2 presents the evaluating criteria based on a 1-9 scale for the pairwise comparison matrix used in this study. For example, considering the following rules i and j .

Rule i : Primary surfaces should be machined prior to secondary surfaces.

Rule j : Planes should always be machined prior to holes and slots.

If Rule i is considered to be much more important than Rule j in the evaluation, a weight of '7' is inserted in the juncture cell (r_{ij}) between Rule i and Rule j . On the contrary, the value of in the juncture cell (r_{ji}) between Rule j and Rule i is set to '1/7'. Based on the evaluating indicator hierarchy, four R-matrices have been built, that is R_1 (3×3), R_{21} (4×4), R_{22} (4×4), and R_{23} (3×3). The value for each element of the four R-matrices is determined based on the experts' experience and knowledge.

2) Calculation of importance weights

There are a number of mathematical techniques to calculate relative importance weights based on the pairwise comparison matrix, such as Eigenvalue, Mean Transformation, or Row Geometric Mean. Considering simplification and implementation on the computer, an approximate method of Eigenvalue approach is used in this research. The process is discussed below.

Step 1: Calculate multiplication (M) of all elements in each row of the pairwise comparison matrix. For the i th row, it is defined as:

$$M_i = \prod_{j=1}^n r_{ij} \quad (7-3)$$

where j is the index of column which elements are in, $j = 1, 2, \dots, n$

i is the index of row which the elements are in, $i = 1, 2, \dots, n$,

and

n is the number of the rows in the pairwise comparison matrix,

which is equal to the number of the columns.

Step 2: Calculate the n -th root of M , that is:

$$\bar{W}_i = \sqrt[n]{M_i} \quad (7-4)$$

where i is the row number in the pairwise comparison matrix, $i = 1, 2, \dots, n$.

Therefore, the relative importance weight vector can be built as the following:

$$\bar{W} = [\bar{W}_1, \bar{W}_2, \dots, \bar{W}_n]^T \quad (7-5)$$

Step 3: Normalise the weight vector, \bar{W}

$$W_i = \frac{\bar{W}_i}{\sum_{j=1}^n \bar{W}_j} \quad (7-6)$$

Thus, the eigenvector for the R-matrix, W can be obtained:

$$W = [W_1, W_2, \dots, W_n]^T \quad (7-7)$$

Step 4: Calculate the maximum Eigenvalue of R-matrix, λ_{\max} :

$$\lambda_{\max} = \sum_{i=1}^n \frac{(RW)_i}{W_i} \quad (7-8)$$

Step 5: Monitor the consistency of the pairwise comparison matrix.

Consistency means that the decision exhibits coherent judgement in specifying the pairwise comparison of the criteria or alternatives [Taha, 1997]. For the pairwise comparison matrix, R-matrix, the consistency can be described in mathematics as the following:

$$r_{ij} r_{jk} = r_{ik}, \quad \text{for all } i, j \text{ and } k \quad (7-9)$$

According to the definition of actually logical meanings for each element in the pairwise comparison matrix (R-matrix), R-matrix should be consistent. Mathematically, for any 2×2 matrix, it is always consistent because the columns of any 2×2 comparison matrix are dependent. However, for all $n \times n$ comparison matrices ($n > 2$), they are usually inconsistent, as the comparison matrix is usually constructed based on human judgement from which some degree of inconsistency is expected. Therefore, the problem of consistency can be converted into determining whether or not a level of consistency is "reasonable".

Assuming $\lambda_1, \lambda_2, \dots, \lambda_n$ are the Eigenvalues of R-matrix, which satisfy the following equation:

$$RW = \lambda W \quad (7-10)$$

According to linear algebra, the following expression is obtained:

$$\sum_{i=1}^n \lambda_i = n \quad (7-11)$$

where $r_{ii} = 1$.

If R-matrix is consistent, then

$$\lambda_1 = \lambda_{\max} = n \quad (7-12)$$

Otherwise,

$$\lambda_1 = \lambda_{\max} > n \quad (7-13)$$

$$\sum_{i=1}^{n-1} \lambda_i = n - \lambda_{\max} \quad (7-14)$$

In this case, the closer λ_{\max} is to n , the more consistent is the comparison matrix. As a measure, an inconsistency ratio (CR) is calculated to monitor the consistency of the comparison matrix:

$$CR = \frac{CI}{RI} \quad (7-15)$$

where CI is a consistency index of R-matrix,

$$CI = \frac{\lambda_{\max} - n}{n - 1} \quad (7-16)$$

and RI is a random consistency index of R-matrix. RI is determined empirically [Golden *et al*, 1989] as the average CI of a large sample of

randomly generated comparison matrices. Table 7.3 gives the values of RI corresponding to n .

Table 7.3. The values of RI

N	1,2	3	4	5	6	7	8
RI	0.00	0.58	0.90	1.12	1.24	1.32	1.41
N	9	10	11	12	13	14	15
RI	1.45	1.49	1.51	1.54	1.56	1.57	1.59

The ratio CR is used to monitor consistency in the following way. If $CR \leq 0.10$, the level of consistency is acceptable. Otherwise, the inconsistency in R -matrix is very high and the decision-maker is advised to check the element r_{ij} of R -matrix to produce a more consistent matrix.

Indeed, consistency of the comparison matrix is important because all other data can be deduced logically from this basic data.

Step 6: Calculate the weights in the lower level. For the lower level, the weights relating to the total objective can be given by the following expression:

$$W_i = \sum_{j=1}^m W_j^0 W_{ij}^s \quad (7-17)$$

where W_i is the importance weight of the i th sub-criteria for the total objective;

W_j^0 is the importance weight of the j th criteria in the higher level for the total objective; and

W_{ij}^s is the importance weight of the i th sub-criteria for the j th criteria in the higher level.

7.2.3 Construction of evaluating matrix

Each rule has a precedence matrix named V-matrix. As mentioned in Chapter 6, each operation pair has a reward/penalty precedence value for a corresponding rule, which is stored in the evaluation library. The reward/penalty precedence scale used is identified as:

A reward: If the precedence in the operation pair (e.g. operation i and operation j) satisfies the manufacturing rule k , then a positive decimal value (reward) is given according to the satisfying degree: $0 < v_{kij} \leq 1$.

A penalty: If the precedence in the operation pair (e.g. operation i and operation j) does not satisfy the manufacturing rule k , then a large positive value (penalty) is given according to the destruction degree: $v_{kij} > 1$.

According to the reward/penalty precedence values, the V-matrix for rule k can be defined as:

$$V_k = \begin{bmatrix} v_{k11} & \cdots & v_{k1i} & \cdots & v_{k1n} \\ \vdots & & \vdots & & \vdots \\ v_{ki1} & \cdots & v_{kii} & \cdots & v_{kin} \\ \vdots & & \vdots & & \vdots \\ v_{kn1} & \cdots & v_{kni} & \cdots & v_{knn} \end{bmatrix} \quad (7-18)$$

where V_k is the V-matrix for rule k ;

$$i = 1, 2, \dots, n;$$

n is the number of machining operations in a group (for operation sequencing) or the number of machining operation groups (for group sequencing);

$k=1, 2, \dots, m$;

m is the number of selected rules, and

v_{kij} is the reward/penalty precedence value for rule k if operation i is prior to operation j (for operation sequencing). For group sequencing, v_{kij} is the reward/penalty precedence value based on rule k if operation s is prior to operation t , while operations s and t are the last operation in group i and the first operation in group j , respectively.

Because the penalty/rewards precedence matrix has considered precedence constraint relationships, it is not necessary to recheck a process sequence after its genetic operators. With the penalty/rewards precedence matrix, the basic optimisation problem is transferred into an alternative formulation, that is, a numerical solution is sought by solving a sequence of unconstrained minimisation problems.

7.2.4 Evaluation result

The synthesis evaluation can be set up based on the following linear addition expression:

$$f_m = \sum_{k=1}^m \sum_{i=1}^n \sum_{\substack{j=1, \\ G_i > G_j}}^n w_k v_{kij} \quad (7-19)$$

where f_m is the degree of satisfaction with process sequence rules,

w_k is the relative weight for the k th constraint rule,

v_{kij} is the reward/penalty precedence value for rule k if operation i is prior to operation j (for operation sequencing) or the reward/penalty precedence value for rule k if group i is prior to group j (for group sequencing), and

$G_i > G_j$ means operation i is arranged before operation j (for operation sequencing) or group i is arranged before group j (for group sequencing).

f_m is a positive decimal value, that is $f_m \geq 0$. The lower the value is, the higher the sequencing satisfies with process sequence rules.

7.3 Evaluation of time and cost

Because detailed information on tool paths and machining parameters have not been determined so far, instead of accurate cost and time, estimation functions are used to calculate a process plan's cost and time.

1) Cost

The cost for a process plan consists of five main elements described below.

- Machine cost (CM)

Machine cost is the total cost of the machines used in an operation.

- Tool cost (CT)

Tool cost is the total cost of the cutting tools used in an operation.

- Machine change cost (CMC)

It includes the total cost spent to change the machine tools occurring when two adjacent operations are performed on different machines.

- Tool change cost (CTC)

Similarly, it is the total cost spent for cutting tool changing occurring when two adjacent operations are performed on the same machine but with different tools.

- Setup change cost (CSC)

It is the total cost for setup changing occurring when two adjacent operations are performed on the same machine but with different setups.

Based on the above elements, the cost for a process sequence can be expressed as:

$$C = \sum_{i=1}^n GC_i + \sum_{\substack{i=1 \\ j=i+1}}^n GCC_{ij} \quad (7-20)$$

$$GC_i = \sum_{p=1}^P (CM_{ip} + CT_{ip}) \quad (7-21)$$

$$GCC_{ij} = CMC_{st} + CTC_{st} + CSC_{st} \quad (7-22)$$

where C is the total cost for the process plan.

GC_i is the machining cost for the i th operation group.

GCC_{ij} is the changing cost for two adjacent groups, the i th group and the j th group. Here, the index of the group represents the process sequence of the group, e.g. the 1st group is processed first, and then the 2nd group, and so on.

CM_{ip} is the machine cost for the p th operation in the i th group.

CT_{ip} is the tool cost for the p th operation in the i th group.

P is the number of the operations in the i th group.

CMC_{st} is the machine change cost for the last operation in i th group to the first operation in j th group.

CTC_{st} is the tool change cost for the last operation in i th group to the first operation in j th group.

CSC_{st} is the setup change cost for the last operation in i th group to the first operation in j th group.

Then, the relative evaluating value for manufacturing cost, f_c , can be gained as:

$$f_c = \frac{C}{C_{\max}} \quad (7-23)$$

where C_{max} is the maximum manufacturing cost for the component that the company can accept. The C_{max} for a component is given by user.

2) Time

Similar to the calculation of manufacturing cost, the time of a process plan includes the following elements:

- Machining time (TM)

Machining time is the total time needed when a component is machined in one operation. It consists of the cutting time, machine idle time due to preparation and idle tool motion, such as loading, unloading, tool approach and depart.

- Machine change time (TMC)

It refers to the time occurring when two adjacent operations are performed on different machines.

- Tool change time (TTC)

It is the time spent on changing tools when two adjacent operations are performed on the same machine but with different tools.

- Setup change time (TSC)

It occurs when two adjacent operations are performed on the same machine but with different setups.

Therefore, the time for a process sequence can be expressed as:

$$T = \sum_{i=1}^n GT_i + \sum_{\substack{i=1 \\ j=i+1}}^n GTC_{ij} \quad (7-24)$$

$$GT_i = \sum_{p=1}^P TM_{ip} \quad (7-25)$$

$$GTC_{ij} = TMC_{st} + TTC_{st} + TSC_{st} \quad (7-26)$$

where T is the total time for the process plan.

GT_i is the machining time for the i th group.

GTC_{ij} is the changing time for two adjacent groups, the i th group and j th group. Here, the index of the group represents the process sequence of the group, e.g. the 1st group is processed first, and then the 2nd group, and so on.

TM_{ip} is the machining time for the p th operation in the i th group.

P is the number of the operations in the i th group.

TMC_{st} is the machine change time from the last operation in i th group to the first operation in j th group.

TTC_{st} is the tool change time from the last operation in i th group to the first operation in j th group.

TSC_{st} is the setup change time from the last operation in i th group to the first operation in j th group.

Then, the relative evaluating value for manufacturing time, f_t , can be gained as:

$$f_t = \frac{T}{T_{max}} \quad (7-27)$$

where T_{max} is the longest manufacturing time for the component that the company can accept. The T_{max} for a component is given by user.

7.4 Weight calculation

Considering the disparities of product types and production conditions, different values of w_m , w_c and w_t can be assigned using an artificial neural network. The following three aspects are discussed to construct the network: input representation, output format, network topology.

7.4.1 Input representation

The relative weights for process sequence rules, manufacturing cost and manufacturing time are always dependent on the component design and technological requirements, and manufacturing circumstances. Thus, three main factors are considered for the input.

1) Complexity of component

It can be difficult to evaluate the complexity of a component because of a number of considerations including:

- The number of features in the component

- The classes of features in the component
- The relationships among features in the component
- The technological and design requirements for the component, such as dimension tolerance, surface roughness
- The material of the component

In the meantime, the evaluation is non-linear and multi-dimensioned. For example, the difficulty of component manufacture increases with the number of features. On the other hand, features do not contribute equally to the manufacturing difficulty. Moreover, all these considerations affect each other. For instance, complex feature relationships may result in high technological requirements. In addition, some vague factors are included, such as geometrical complexity and material, so that expert knowledge and experience are needed. Fuzzy evaluation techniques are employed with the neural network to tackle such complicated evaluation of a component. The proposed fuzzy evaluation (to be discussed further in the next section) focuses on evaluating the complexity of a feature according to its geometrical complexity, technological requirements and machining capability. Table 7.4 shows five levels of feature complexity.

Table 7.4. Evaluation for feature complexity

Evaluating value	Description	Type
<0.25	Very easy to be manufactured	Type I
0.25-0.5	Easy to be manufactured	Type II
0.5-0.75	Not difficult to be manufactured	Type III
0.75-1	Difficult to be manufactured	Type IV
≥1	Very difficult to be manufactured	Type V

Based on the output of the fuzzy evaluation of features, 20 input neurons are designed, of which each four neurons represent a binary number, detailed below.

Neurons 1-4: the number of features in the component, which are very easy to manufacture (Type I).

Neurons 5-8: the number of features in the component, which are easy to manufacture (Type II).

Neurons 9-12: the number of features in the component, which are not difficult to manufacture (Type III).

Neurons 13-16: the number of features in the component, which are difficult to manufacture (Type IV).

Neurons 17-20: the number of features in the component, which are very difficult to manufacture (Type V).

For example, if a component has 10 features and 5, 1, 3 and 1 features belong to Type I, Type II, Type III and Type IV, respectively, the values for the 20 input neurons are shown in Table 7.5.

Table 7.5. An example of input neurons 1 to 20

Input neurons 1→20																			
0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0
Type I				Type II				Type III				Type IV				Type V			
5				1				3				1				0			

2) Production batch size

In practice, the process plan of a component for mass production, medium production, small production or single production may be different significantly because of different production objectives. For example, in mass production, the process plan is repeated many times with a new workpiece. Because production batch size has a great effect on process planning, it is taken as a crucial consideration for the input. The evaluating values of the production batch size are allocated in Table 7.6.

Table 7.6. The evaluating values for production batch size

Production batch size	Quantity	Evaluating value
Single production	<20	1
Small production	20-200	0.6
Medium production	201-5000	0.3
Mass production	>5000	0

3) Production urgency

An indicator is designed to represent how urgent the component is needed. The evaluating values are allocated in Table 7.7. The production urgency indicator is an important parameter affecting the relative importance of timing.

Table 7.7. The evaluating values for production urgency

Production urgency	Time (days)	Evaluating value
Very urgent	<7	1
Urgent	7-14	0.6
Normal	15-30	0.3
Not urgent	>30	0

7.4.2 Output format

According to the problem to be solved, the output consists of three neurons, which represent w_m , w_c and w_t , respectively. A real number between 0 and 1 is assigned to them. As relative weights, w_m , w_c , and w_t , are normalised before they are input into the final fitness calculation, that is $w_m+w_c+w_t=1$.

7.4.3 Topology and the training method of the neural network

The proposed neural network uses a typical three-layer BP (back-propagation) structure, consisting of an input layer, a hidden layer and an output layer. There are 22 neurons and 3 neurons in the input layer and the output layer, respectively. The number of neurons in the hidden layer is a result of experiments using various architectures, and a hidden layer of 10 neurons has proved to be the most 'appropriate'.

7.5 Fuzzy evaluation of feature complexity

Based on the above characteristics of feature complexity, a feature evaluation method is proposed using fuzzy mathematics.

7.5.1 Fuzzy evaluation model

Based on the theory of fuzzy mathematics for synthesis evaluation, an analytical model is established for evaluating feature complexity. The model is described below.

Assume that the domain of the evaluation factors is A :

$$A = \{a_1, a_2, \dots, a_i, \dots, a_n\} \quad (7-28)$$

where a_i represents the factor of evaluation, and $i=1, 2, 3, \dots, n$.

The domain of the evaluation grades is V :

$$V = \{v_1, v_2, \dots, v_j, \dots, v_m\} \quad (7-29)$$

where v_j expresses the evaluation results of complex degree for each factor, and $j=1, 2, 3, \dots, m$. The current method adopts five-grades: very simple, simple, general, complex, very complex. That is,

$$V = \{\text{very simple, simple, general, complex, very complex}\},$$

Accordingly, a vector with five values is defined as:

$$\bar{V} = \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \\ \bar{v}_4 \\ \bar{v}_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.25 \\ 0.5 \\ 0.75 \\ 1 \end{bmatrix} \quad (7-30)$$

Supposing a fuzzy evaluation matrix U can be established by:

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1j} & \cdots & u_{15} \\ \vdots & & \vdots & & \vdots \\ u_{i1} & \cdots & u_{ij} & \cdots & u_{i5} \\ \vdots & & \vdots & & \vdots \\ u_{n1} & \cdots & u_{nj} & \cdots & u_{n5} \end{bmatrix} \quad (7-31)$$

where u_{ij} indicates the evaluation value of the i th evaluation factor, a_i to the membership degree of the j th evaluation grade, v_j .

Then, a fuzzy vector \bar{U} can be calculated as:

$$\bar{U} = \begin{bmatrix} \bar{u}_1 \\ \vdots \\ \bar{u}_i \\ \vdots \\ \bar{u}_n \end{bmatrix} = U\bar{V} = \begin{bmatrix} \sum_{j=1}^5 (u_{1j} \times \bar{v}_j) \\ \vdots \\ \sum_{j=1}^5 (u_{ij} \times \bar{v}_j) \\ \vdots \\ \sum_{j=1}^5 (u_{nj} \times \bar{v}_j) \end{bmatrix} \quad (7-32)$$

In order to consider the influence of interactions among the evaluation factors, a weight is introduced for performing the synthesis evaluation on each factor. The fuzzy set of weights, W is normalised:

$$W = (w_1, w_2, \dots, w_i, \dots, w_n) \quad (7-33)$$

where w_i denotes the corresponding weight of the i th factor, a_i , and

$$\sum_{i=1}^n w_i = 1 \quad (7-34)$$

The Analytical Hierarchical Process is also used to identify the weights based on the expert knowledge.

Finally, the fuzzy synthesis evaluation can be performed with the fuzzy operation

$$Y = W \bullet \bar{U} = \max(w_1 \times \bar{u}_1, w_2 \times \bar{u}_2, \dots, w_5 \times \bar{u}_5) \quad (7-35)$$

7.5.2 Feature evaluation

Because different feature class has different geometrical parameters and different technological indicators, it is not easy to obtain a unique and efficient evaluating matrix to satisfy various features. At the same time, because there is a number of machining features on one component and feature classes are varied, the evaluation matrix will become larger and the identification of weights will become inefficient. Additionally, the evaluation matrix must be rebuilt when a new feature category is added. Thus a feature evaluation method is structured, which consists of a control interface and several feature evaluators. The control interface is used for recognising the feature class and processing it, then putting it into the corresponding feature evaluator, which is used for evaluating the complexity for specifically feature class. All feature evaluators are separately built, which can improve its efficiency and effective. When a new feature category is added on a component, the corresponding feature evaluator is added or modified but the other feature evaluators need not be changed. The integrated architecture of this evaluation method is shown in Figure 7.3.

For the objective of evaluation for feature complexity, the following factors are considered.

1) Feature class

Feature class is about typical geometrical characteristics, which is one of major factors that determine the feature complexity. The corresponding grade memberships are given with expert knowledge. For example, Table 7.8 shows the grades of round holes.

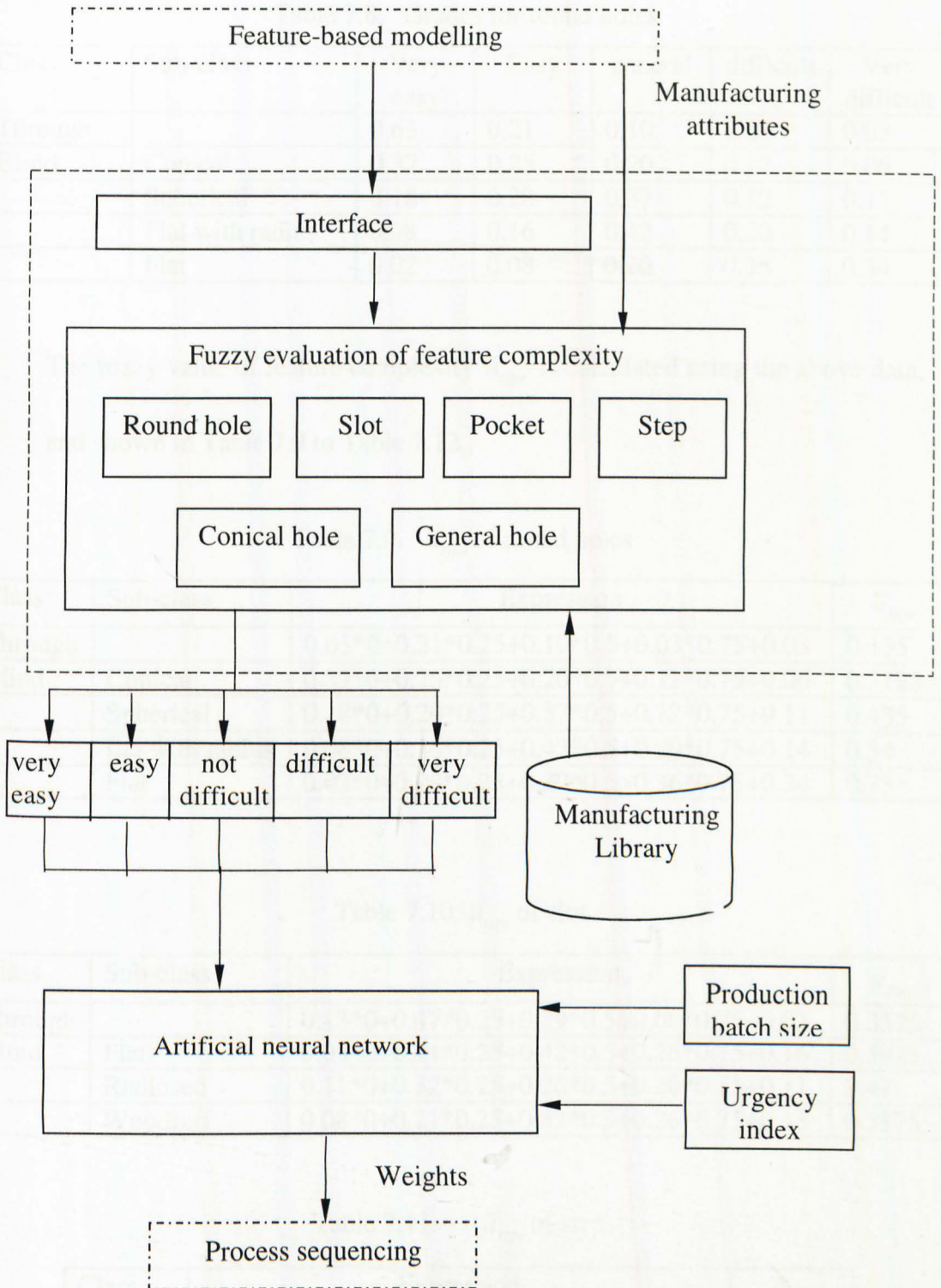


Figure 7.3 The structure of weights collated

Table 7.8. Grades for round holes

Class	Sub-class	Very easy	Easy	general	difficult	Very difficult
Through		0.63	0.21	0.10	0.03	0.03
Blind	Conical	0.37	0.25	0.20	0.12	0.06
	Spherical	0.18	0.20	0.37	0.12	0.11
	Flat with radius	0.08	0.16	0.42	0.20	0.14
	Flat	0.02	0.08	0.20	0.36	0.34

The fuzzy value of feature complexity \bar{u}_{type} is calculated using the above data, and shown in Table 7.9 to Table 7.12.

Table 7.9. \bar{u}_{type} of round holes

Class	Sub-class	Expression	\bar{u}_{type}
Through		$0.63*0+0.21*0.25+0.10*0.5+0.03*0.75+0.03$	0.155
Blind	Conical	$0.37*0+0.25*0.25+0.20*0.5+0.12*0.75+0.06$	0.3125
	Spherical	$0.18*0+0.20*0.25+0.37*0.5+0.12*0.75+0.11$	0.435
	flat with radius	$0.08*0+0.16*0.25+0.42*0.5+0.20*0.75+0.14$	0.54
	Flat	$0.02*0+0.08*0.25+0.20*0.5+0.36*0.75+0.34$	0.73

Table 7.10. \bar{u}_{type} of slot

Class	Sub-class	Expression	\bar{u}_{type}
Through		$0.13*0+0.47*0.25+0.29*0.5+0.08*0.75+0.03$	0.3525
Blind	Flat	$0.05*0+0.11*0.25+0.42*0.5+0.26*0.75+0.16$	0.5925
	Radiused	$0.11*0+0.32*0.25+0.26*0.5+0.20*0.75+0.11$	0.47
	Woodraff	$0.08*0+0.21*0.25+0.32*0.5+0.26*0.75+0.13$	0.5375

Table 7.11. \bar{u}_{type} of step

Class	Expression	\bar{u}_{type}
Through	$0.21*0+0.45*0.25+0.26*0.5+0.05*0.75+0.03$	0.31
Blind	$0.08*0+0.39*0.25+0.45*0.5+0.05*0.75+0.03$	0.39

Table 7.12. \bar{u}_{type} of pocket

Class	Sub-class	Expression	\bar{u}_{type}
Open pocket		$0.08*0+0.13*0.25+0.37*0.5+0.31*0.75+0.11$	0.56
Closed pocket	Polygon closed pocket	$0.11*0+0.16*0.25+0.21*0.5+0.29*0.75+0.23$	0.5925
	Part-circle closed pocket	$0.03*0+0.08*0.25+0.26*0.5+0.29*0.75+0.34$	0.7075
	Double-semi-circle closed pocket	$0.16*0+0.23*0.25+0.34*0.5+0.19*0.75+0.08$	0.45

2) Nominal dimensions

Dimensions are a very important factor for process planning because they not only relate to the machining time and tool changes, but also constraint the accessibility and fixturing of the feature.

- For round holes, there are two nominal dimensions: diameter and depth and there is a related factor: the ratio between depth and diameter, depth/diameter (Figure 7.4).
- For slots, there are depth, length, width and character angle (defined in Chapter 3).
- For pockets, the nominal dimensions include depth, minimum edge length, maximum edge length, maximum angle between faces, and minimum angle between faces.
- For steps, the nominal dimensions include length, depth and width.

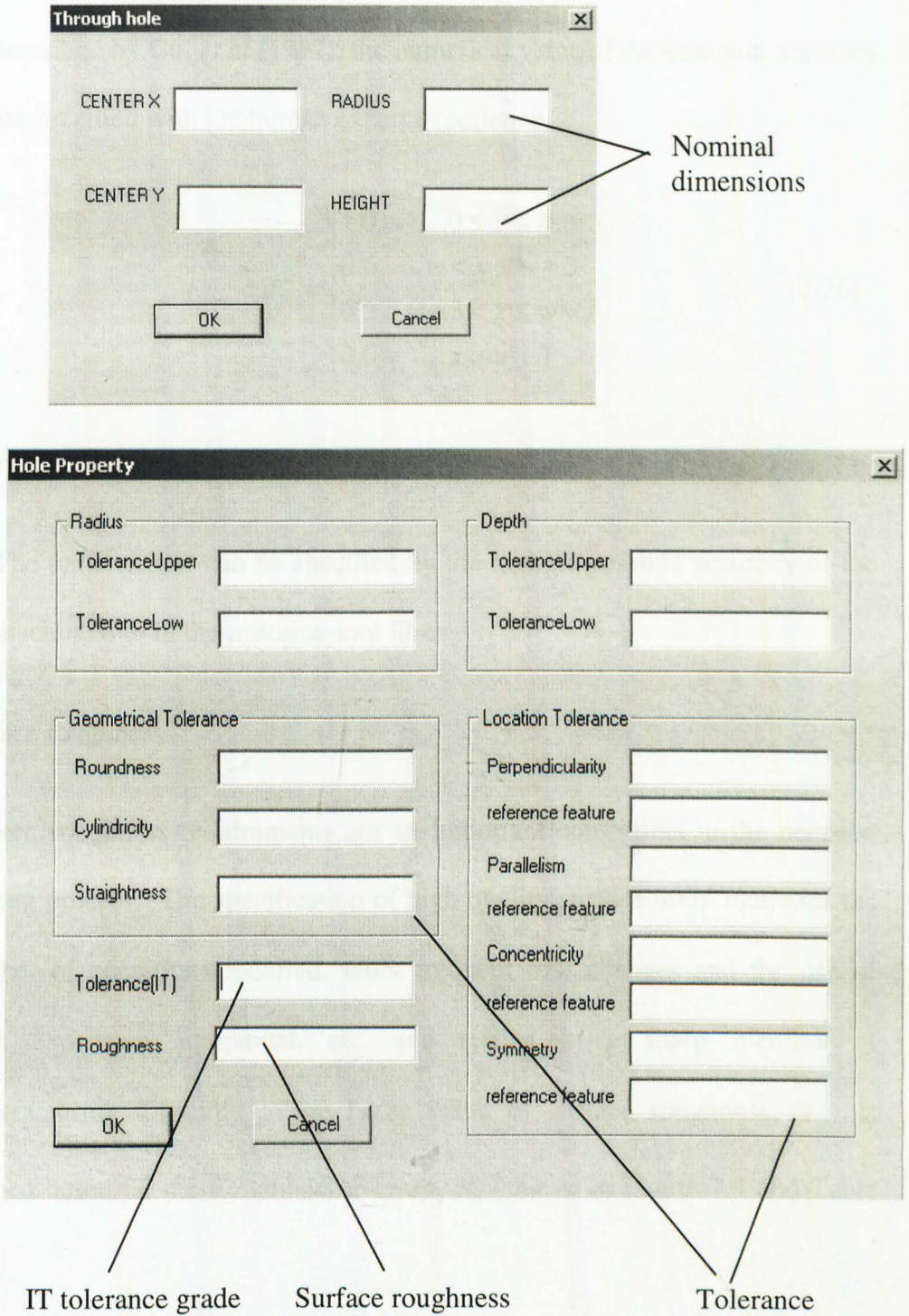


Figure 7.4 Examples of nominal dimensions, dimensional accuracy, surface roughness and tolerances

3) Dimensional accuracy

As described by Gu, *et al* [1997], the numerical value of dimensional accuracy can be fuzzified with the human expert experience.

$$\bar{u}_x^w = \begin{cases} 1.0 & 0 \leq x \leq w \\ 0.6 & w < x \leq w+3 \\ 0.3 & w+3 < x \leq w+7 \\ 0.0 & x > w+7 \end{cases} \quad (7-36)$$

where x is the IT tolerance grade (Figure 7.4).

The parameter w can be specified by the highest possible accuracy of the machine tool in the machine-tool library.

4) Surface roughness

Surface roughness requirements are an important constraints in the decision making process. The specification of high quality surface often increases the number of operations required, more frequent tool changes and the use of more expensive equipment, etc, and results in a sharp increase of manufacturing difficulty. The fuzzy value of surface roughness, \bar{u}_{sr} , is defined based on the Roughness average, R_a (shown in Figure 7.4 and Table 7.13)

Table 7.13. Fuzzy value for surface roughness

Machining type	$R_a(\mu\text{m})$	\bar{u}_{sr}
Rough machining	100-12.5	0
Semi-finish machining	6.3-1.6	0.3
Finish machining	0.80-0.20	0.6
Fine-finish machining	0.100-0.006	1

5) Material

The present method uses the hardness of material to represent its characteristics. The fuzzy evaluating value of material \bar{u}_m can be calculated with the function for specific cutting force k_c (N/mm^2) on the workpiece

$$\bar{u}_m = \frac{(k_c - K_{c\min})}{(K_{c\max} - K_{c\min})} \quad (7-37)$$

where $K_{c\min}$ and $K_{c\max}$ are the minimum and maximum values of k_c , stored in the material library.

6) Tolerance

Tolerance also dominates the manufacturing difficulty. In terms of design, there are six geometric tolerances and eight location tolerances. However, not all tolerances are necessary to specify a feature, i.e. dependent on the feature class.

- Round hole: straightness, roundness, cylindricity, parallelism, perpendicularity, concentricity and symmetry (Figure 7.4).
- Slot: flatness, parallelism, perpendicularity and symmetry.
- Pocket: flatness, perpendicularity and symmetry.
- Step: flatness, parallelism, perpendicularity and symmetry.

7) Feature relationships

Feature relationships can affect process planning seriously, especially process sequence. For example, the parent feature should be machined prior to its child feature, or it may be difficult to access the child feature, or a special tool or special fixture may be required, or it is even impossible to machine the child feature. Based on the relationships defined, three cases are considered: parent feature, child feature and connect feature. The value \bar{u}_{fr} is determined below

$$\bar{u}_{fr} = \frac{N_{parent} + N_{child} + N_{connect}}{N_{max}} \quad (7-38)$$

where N_{parent} is the number that the feature is regarded as a parent feature.

N_{child} is the number that the feature is regarded as a child feature.

$N_{connect}$ is the number that the feature has a connect relationship with other features.

N_{max} is the maximum number of relationships between the feature and other features, which is currently set to 8.

Based on the above analysis, the domain of the evaluation factors for each feature class is determined.

1) for round hole, $A = \{a_1, a_2, \dots, a_i, \dots, a_{15}\}$

2) for slot, $A = \{a_1, a_2, \dots, a_i, \dots, a_{13}\}$

3) for pocket, $A = \{a_1, a_2, \dots, a_i, \dots, a_{13}\}$

4) for step, $A = \{a_1, a_2, \dots, a_i, \dots, a_{12}\}$

The various relative weights are calculated with the AHP, which has been described before. Finally, the result for feature evaluation is calculated in the fuzzy synthesis evaluation described at the end of Section 7.5.1 (i.e. $Y = W \bullet \bar{U} = \max(w_1 \times \bar{u}_1, w_2 \times \bar{u}_2, \dots, w_5 \times \bar{u}_5)$). Figure 7.6 shows the flow of the evaluation results fed into the artificial neural network with weights allocated.

7.6 Summary

This chapter has described the optimisation strategy developed for process sequencing. The most important characteristic is that the strategy is based on multi-objective fitness: minimum manufacturing cost, shortest manufacturing time and best satisfaction of process sequence rules, which previous research does not consider. A hybrid approach is employed to incorporate genetic algorithm and fuzzy analysis techniques for process sequencing. After a brief introduction of GA, the proposed GA is discussed covering encoding scheme, initialisation, genetic operators, fitness function and stop criteria. Four key issues are discussed to further explain fitness function. Firstly, the analytical hierarchical process is proposed to evaluate the satisfaction degree of process sequence rules. AHP not only transfers the manufacturing sequence constraints into a numerical solution, but also makes the evaluation more flexible and more adaptive. Then, two functions to calculate manufacturing cost and time are designed. Thirdly, relative

weights are allocated for the three major factors, process sequence rules, manufacturing cost and time is presented. Finally, an evaluation for feature complexity is solved with fuzzy techniques.

8.1 Facilities

A prototype system has been implemented for the proposed work with the following facilities:

- 1) Computer: PIII 500 PC, 8.4GB HD, 128Mb Memory
- 2) Operating system: Windows 2000
- 3) ACIS geometric modeling, version 1.0
- 4) Visual C++ 6.0
- 5) Visual Basic 6.0
- 6) Microsoft Access 97
- 7) The MATLAB neural network toolbox (Release 1.0)

8.2 System Implementation

The block diagram of the prototype system is shown in Figure 8.1. It consists of the following four major phases:

Chapter 8

Implementation and Testing of Prototype System

8.1 Facilities

A prototype system has been implemented for the proposed work with the following facilities:

- 1) Computer: PIII 500 PC, 8.4Gb HD, 128Mb Memory
- 2) Operating system: Windows 2000
- 3) ACIS geometric modeller version 7.0
- 4) Visual C++ 6.0
- 5) Visual Basic 6.0
- 6) Microsoft Access 97
- 7) The MATLAB neural network toolbox (Release 12)

8.2 System implementation

The block diagram of the prototype system is shown in Figure 8.1. It consists of the following four major phases.

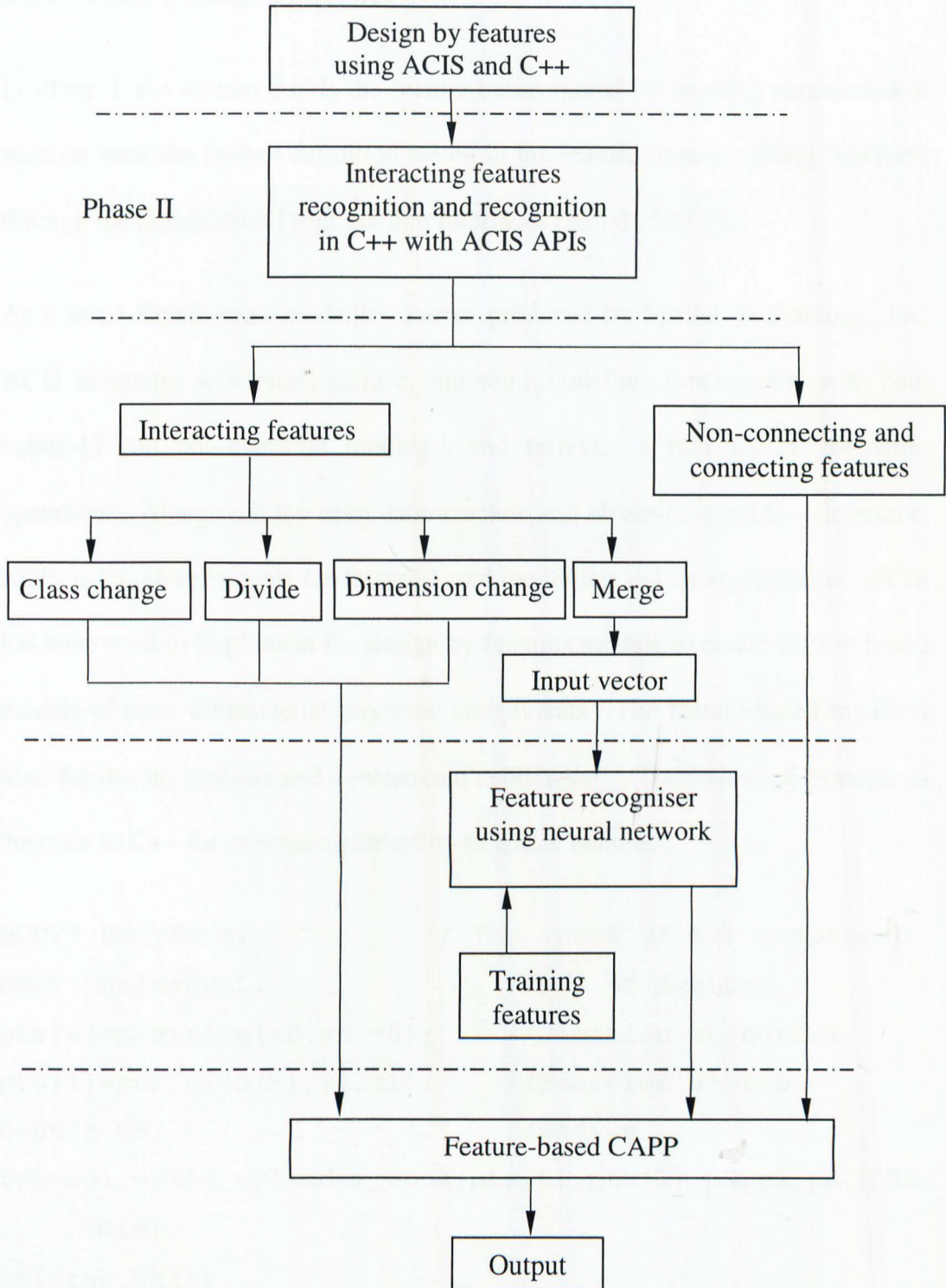


Figure 8.1 The block diagram of the prototype system

8.2.1 Phase I: Design by features

In Phase I, the system builds the feature-based model by creating parameterised features with the feature definition stored in the feature library, calling functions through the Application Programming Interfaces (APIs) of ACIS.

As a three-dimensional modelling kernel produced by Spatial Technology, Inc, ACIS integrates wireframe, surface, and solid modelling functionality with both manifold and non-manifold topology, and provides a rich set of geometric operations. Along with the open data structure and object-oriented C++ interface, ACIS is therefore suitable for feature-based modelling and its applications. ACIS has been used to implement the design by features module to create feature-based models of three-dimensional prismatic components. The feature-based model is used for design analysis and downstream applications. The following example is the code in C++ for generating the entity of a hole feature.

```

BODY* BodySave;          // The stock of the component
BODY* hole=NULL;        //SVE of feature
pts[0]=position(x0,y0,z0); //Position at bottom
pts[1]=position(x1,y1,z1); //Position at top
double pR;              //Radius
res=api_solid_cylinder_cone(pts[0],pts[1],pR,pR,pR,NULL
    ,hole);
if(!res.ok())
{
    AfxMessageBox(_T("Error Making hole"));
    exit(1);
}
res=api_subtract(hole, BodySave);

```

Figure 8.2 shows the interface of the design by features module and Figure 8.3 gives an example of the interface for a parameterised feature class.

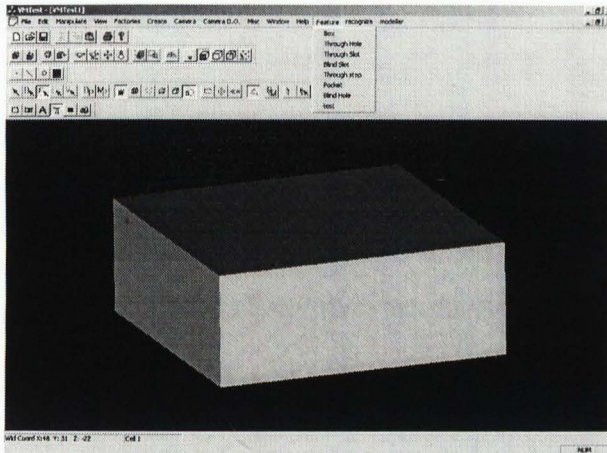


Figure 8.2 The design by features interface

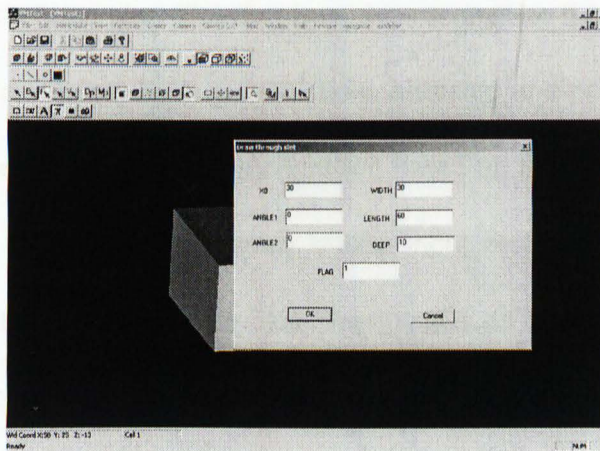


Figure 8.3 The interface for input of feature parameters for a Through Slot

8.2.2 Phase II: Interacting features recognition

In phase II, the system takes the advantage of ACIS to implement the algorithm (detailed in Chapter 4) for searching and checking the interacting features based on Boolean operations. The process is repeated until all the interacting features

are tackled considering the conditions of merge, class change, divide, dimension change and connecting. The implementation is also done in C++ with ACIS APIs. Examples of the API calls include `api_intersect()`, `api_unite()`, `api_subtract()`, and `api_apply_transf()`.

8.2.3 Phase III: ANN-based feature recogniser

Phase III deals specifically with the interacting features to be merged. It is divided into two parts. The first part involves neural network training with input-output sets of training features. The MATLAB neural network toolbox (Release 12), a useful tool to develop neural networks provided by Mathworks is used.

As described as Chapter 5, ANN-based feature recogniser is designed for a three-level hierarchical architecture. The first level is to recognise five primitive feature classes: round hole, conical hole, general hole, slot/step and pocket. The second and third levels are used for further recognition based on the first level for CAPP. For the first level, the input (F-adjacency Matrix input vector) and output matrices of the training features are stored in files "inputtest01.txt" and "outputtest01.txt". Table 8.1 shows some examples of training features.

The input data P and the output data T are loaded as follows:

```
P=load('inputtest01.txt');
T=load('outputtest01.txt');
P=ctranspose(P);
T=ctranspose(T);
```


The code for specifying the size of input matrix is given as:

```
minmax(P);
```

Table 8.1 Examples of training features

Input	Output
1 00 000 000 000 00 0	1 0000
1 30 006 000 000 00 0	1 0000
1 10 005 000 000 00 0	1 0000
1 40 003 000 000 00 0	1 0000
3 00 000 000 000 00 0	0 1000
3 40 006 000 000 00 0	0 1000
6 22 006 300 600 00 0	00 100
6 22 006 400 600 00 0	00 100
6 23 006 200 600 00 0	00 100
6 24 006 200 600 00 0	00 100
6 39 006 300 600 00 0	000 10
6 20 006 000 000 00 0	000 10
6 30 006 000 000 00 0	000 10
6 40 006 000 000 00 0	000 10
6 32 306 230 630 60 0	0000 1
6 23 306 230 630 60 0	0000 1
6 22 306 330 630 60 0	0000 1
6 42 306 230 630 60 0	0000 1
6 30 306 430 630 60 0	000 10
6 40 306 430 630 60 0	000 10
6 40 306 330 630 60 0	000 10
6 40 306 230 630 60 0	000 10

Based on the MATLAB command of 'newff', the corresponding multi-layer feedforward neural network architecture is built. For example, the following command creates a three-layer neural network. There are 17 neurons in the hidden layer and five neurons in the output layer. The transfer function is tan-sigmoid (Bipolar sigmoid function) for both the hidden output layers. The training function is `traincgp`, which is Polak-Ribiere Update.

```
net=newff(minmax(P),[17,5],{'tansig','tansig'},'traincgp');
```

Then training function is used to minimise the error and update the weights using the following code:

```
[net,tr]=train(net,P,T);
```

After the training process, the final weights and biases are stored in two files, respectively. For example, for the first level, the two files are "featurefaw.txt" and "featurefab.txt".

The second part uses the trained network to recognise and classify new input patterns or matrices as specific features. The simulating process is run in C++ using the trained network. The F-adjacent vector and V-adjacent vector (described in Chapter 5) are input to the network for the feature pair to be merged. Finally, according to the weights and biases obtained from the training, the final feature class is identified.

8.2.4 Phase IV: CAPP

Phase IV carries out the tasks for CAPP, which include selecting machining operations, identifying feature precedence relations, grouping machining operations and process sequencing. Details have been described in Chapters 6 and 7. Visual Basic, MATLAB neural network toolbox and Microsoft Access are used to implement the feature-based CAPP functions. A design data file (Design.mdb) and manufacturing data file (Manufacturing library.mdb) have been built with Microsoft Access. Design.mdb consists of nine tables, which are listed in Table 8.2. For Manufacturing library.mdb (Table 8.3) has eighteen tables to store information for the manufacturing environment (e.g. materials, machining operations and accuracy).

Table 8.2 Design.mdb

Table name	Information stored
partdata	General information of the component, such as material, component name, component code and production batch
featuredata	Features belong to the component, including all dimensions, tolerances and features' relationships
partnumber	Code of the component
fdatamerge	Merged features
featurelist	Sequenced features list
tprocess	Processes selected for the features
tgroup	Machining groups
rulegroup	Evaluating information of the machining groups for the corresponding manufacturing rules
lastvalue	Evaluation information for process planning for the component

Table 8.3 Manufacturing library.mdb

Table name	Information stored
featureprocessplan	Process planning generated
setupdata	Setups, including machine tools, cutting tools, operation type, accuracy, roughness
Material	Information of materials, such as the hardness, code and name
Cutting speed	Speed for all operations
Stock	Types of stock
Machine tools	Machine tools that can be provided
Cutters	Cutting tools that can be provided
Accuracy	Information of accuracy
urgencytable	Production urgency index
roughevaluation	Evaluation for roughness
setupchangecost	Costs due to setup changes
processchange	Costs due to process changes
featureweight	Weights for evaluating feature complexity
holeevaluation	Evaluating information for feature class of hole
slotevaluation	Evaluating information for feature class of slot
stepevaluation	Evaluating information for feature class of step
pocketevaluation	Evaluating information for feature class of pocket
mtweight	Weights for evaluating process according to manufacturing rules

An ActiveX control for the implementation of a genetic algorithm developed by Jeff Goslin, Xgenetic [Goslin, 2000], is used to assist the implementation of machining operations sequence. A new population of genomes is created by the following code:

```

XGenetic.GenomeLength = lengthg //The number of bits
                                of a chromosome
XGenetic.RangeMaximum = maxrang //The last bit of
                                chromosome
XGenetic.RangeMinimum = minrang //The first bit of
                                chromosome
XGenetic.PopulationSize = pSize //The total number
                                in the initial
                                population
XGenetic.Create

```

Based on the command of evolution of an existing set of genomes:

```
XGenetic.Evolve
```

An extra modified (crossover described in Chapter 7) is added and executed in Visual Basic.

8.3 File management

The system output is a feature-based model and its corresponding process plan.

The feature-based model contains the information of all machining features defining the component, such as geometry, feature class, dimensions and feature relationships. This information is written to four output files: featurerresult.sat, lastresult.txt, save1final.txt and testfinal.txt. The process plan about machining, such as operation type and operations sequence, is stored in the table finalprocessplan in design.mdb.

8.4 Examples

A number of components have been tested with the prototype system. This section presents typical examples to demonstrate the capabilities of the system.

8.4.1 Example 1

This component has been designed in the procedure shown in Figure 8.4. Figure 8.5 shows the tolerance, surface finish of the features.

Feature 1 (Through Step) \cap Feature 2 (Blind Slot):

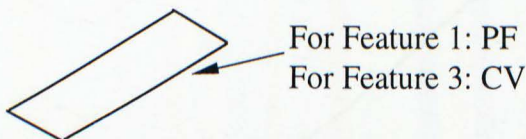
The interacting entity is a face, which is of the PF-CV type:



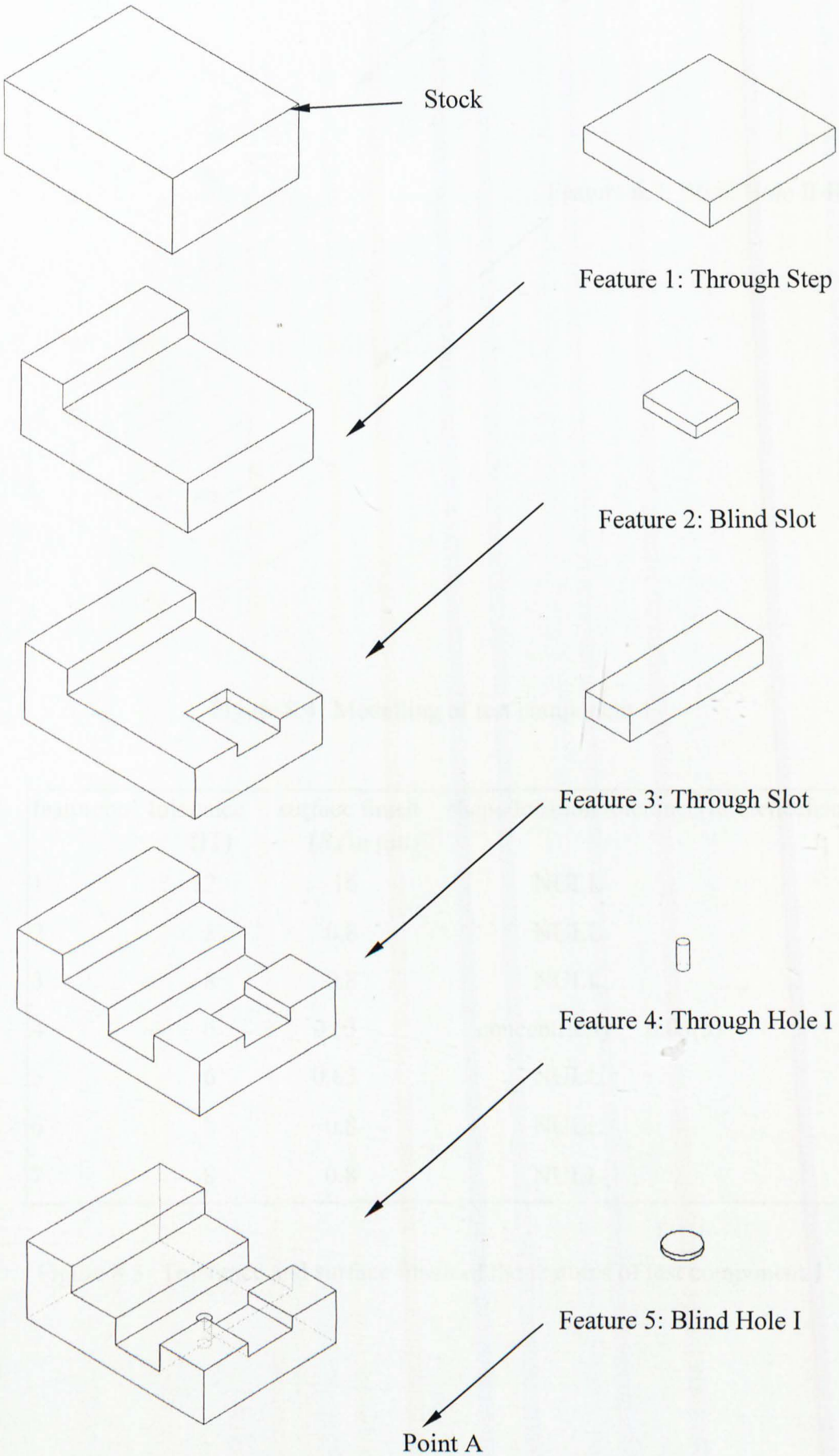
There is a parent-child relationship between Feature 1 (parent feature) and Feature 2 (child feature).

Feature 1 (Through Step) \cap Feature 3 (Through Slot):

The interacting entity is a face, which is of the PF-CV type:



There is a parent-child relationship between Feature 1 (parent feature) and Feature 3 (child feature).



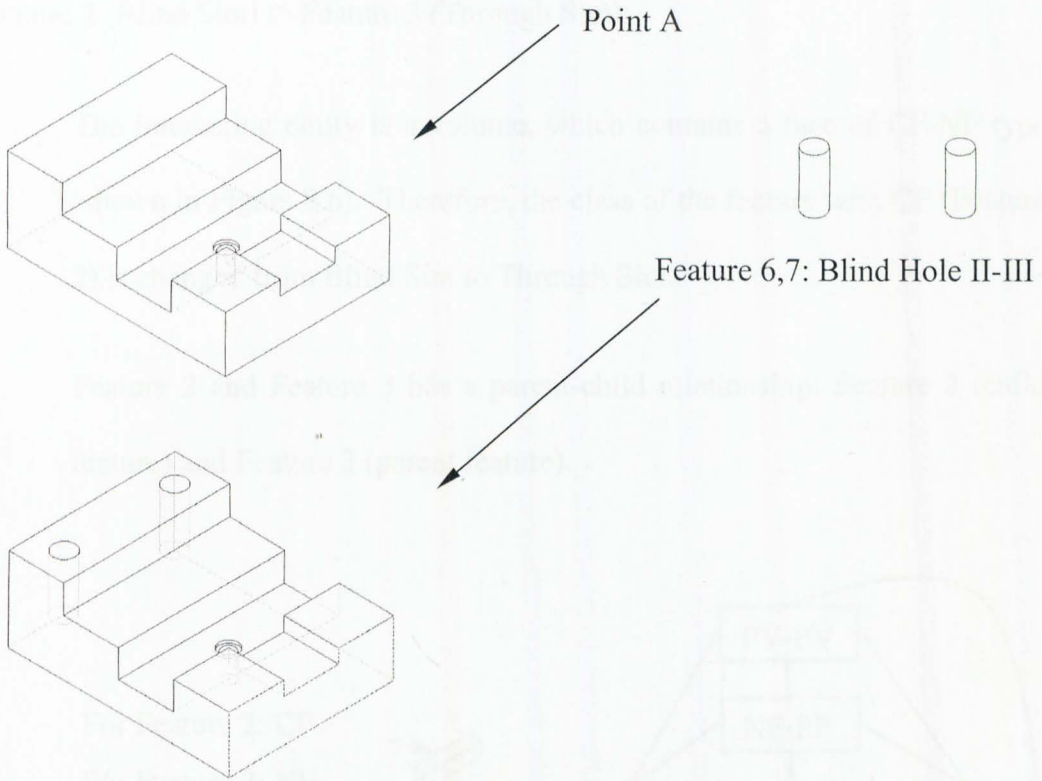


Figure 8.4 Modelling of test component 1

feature no	tolerance (IT)	surface finish (R_a in μm)	shape/location tolerance(reference feature)
1	12	16	NULL
2	8	0.8	NULL
3	8	0.8	NULL
4	6	0.63	concentricity 0.03(5)
5	6	0.63	NULL
6	8	0.8	NULL
7	8	0.8	NULL

Figure 8.5 Tolerance and surface finish of the features of test component 1

Feature 2 (Blind Slot) \cap Feature 3 (Through Slot):

The interacting entity is a volume, which contains a face of CF-NF type (shown in Figure 8.6). Therefore, the class of the feature with CF (Feature 2) is changed from Blind Slot to Through Slot.

Feature 2 and Feature 3 has a parent-child relationship: Feature 2 (child feature) and Feature 3 (parent feature).

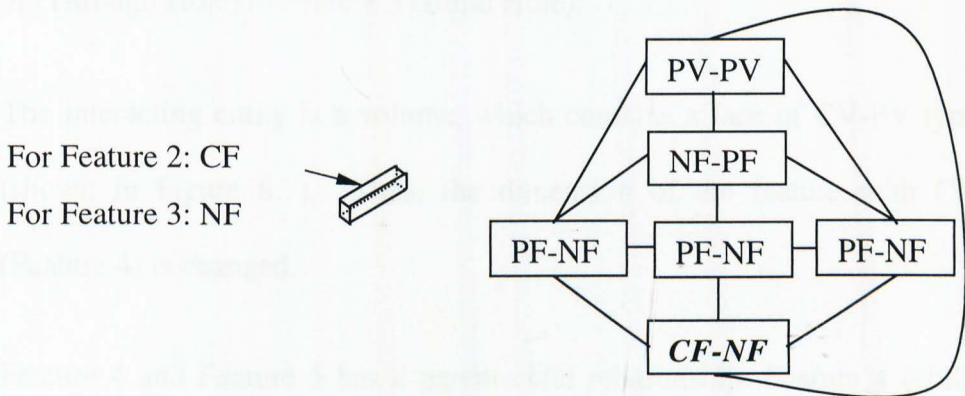
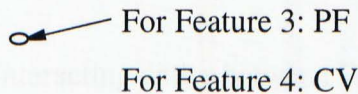


Figure 8.6 Interacting entity between Feature 2 and Feature 3

Feature 3 (Through Slot) \cap Feature 4 (Through Hole):

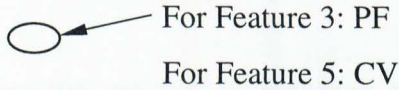
The interacting entity is a face, which type is PF-CV:



It is a parent-child relationship between Feature 3 (parent feature) and Feature 4 (child feature).

Feature 3 (Through Slot) \cap Feature 5 (Blind Hole):

The interacting entity is a face, which type is PF-CV:



It is a parent-child relationship between Feature 3 (parent feature) and Feature 5 (child feature).

Feature 4 (Through Hole) \cap Feature 5 (Blind Hole):

The interacting entity is a volume, which contains a face of CV-PV type (shown in Figure 8.7). Thus, the dimension of the feature with CV (Feature 4) is changed.

Feature 4 and Feature 5 has a parent-child relationship: Feature 4 (child feature) and Feature 5 (parent feature).

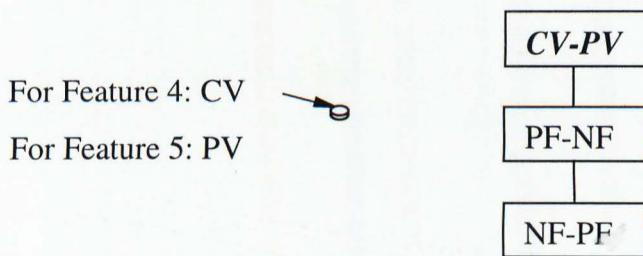


Figure 8.7 Interacting entity between Feature 4 and Feature 5

Figure 8.8 shows the feature recognition result of this example. It can be seen that the proposed heuristic algorithm analyses the Interacting Entity between each feature pair instead the new volume created by all interacting features, making the process simpler. For example, the system only needs to analyse a face (e.g.

Feature 1 and Feature 2) and a volume (e.g. Feature 2 and Feature 3) and does not need complex computation to decompose and reunite edges and faces.

The feature precedence list, machining operation groups and process planning of this example are given in Figure 8.9, Figure 8.10 and Figure 8.11. From the result, it can be seen that a suitable process plan has been produced. For example, Features 6 and 7, which have the same TAD and constraints, are grouped; features of the same class: Features 4 and 5, are to be manufactured successively because they have the same TAD and the parent-child relationship; and Feature 3 is machined prior to Features 2, 4 and 5.

the 1 feature

the original feature class is Thought Step

the last feature class is Thought Step

the feature is the parent feature of feature 2

the feature is the parent feature of feature 3

the 2 feature

the original feature class is Blind Slot

the last feature class is Thought Slot

the feature is the child feature of feature 1

the feature has been changed feature class and become child feature of the 3 feature

the 3 feature

the original feature class is Thought Slot

the last feature class is Thought Slot

the feature is the child feature of feature 1

the feature has changed the 2 feature's class and is parent feature of the 2 feature

the feature is the parent feature of feature 4

the feature is the parent feature of feature 5

the 4 feature

the original feature class is Through Hole

the last feature class is Through Hole

the feature is the child feature of feature 3

the dimension of feature is changed and become child feature of the 5 feature

the 5 feature

the original feature class is Blind Hole

the last feature class is Blind Hole

the feature is the child feature of feature 3

the feature changed the dimension of the 4 feature and is parent feature of the 4 feature

the 6 feature

the original feature class is Blind Hole

the last feature class is Blind Hole

the 7 feature

the original feature class is Blind Hole

the last feature class is Blind Hole

Figure 8.8 Result of feature recognition for test component 1

Index	featureno	typeno	flagno
1	1	Through Step	1
2	3	Blind Slot	1
3	2	Through Slot	1
4	5	Blind Hole	1
5	4	Through Hole	1
6	6	Blind Hole	0
7	7	Blind Hole	0

Figure 8.9 Feature precedence list of test component 1

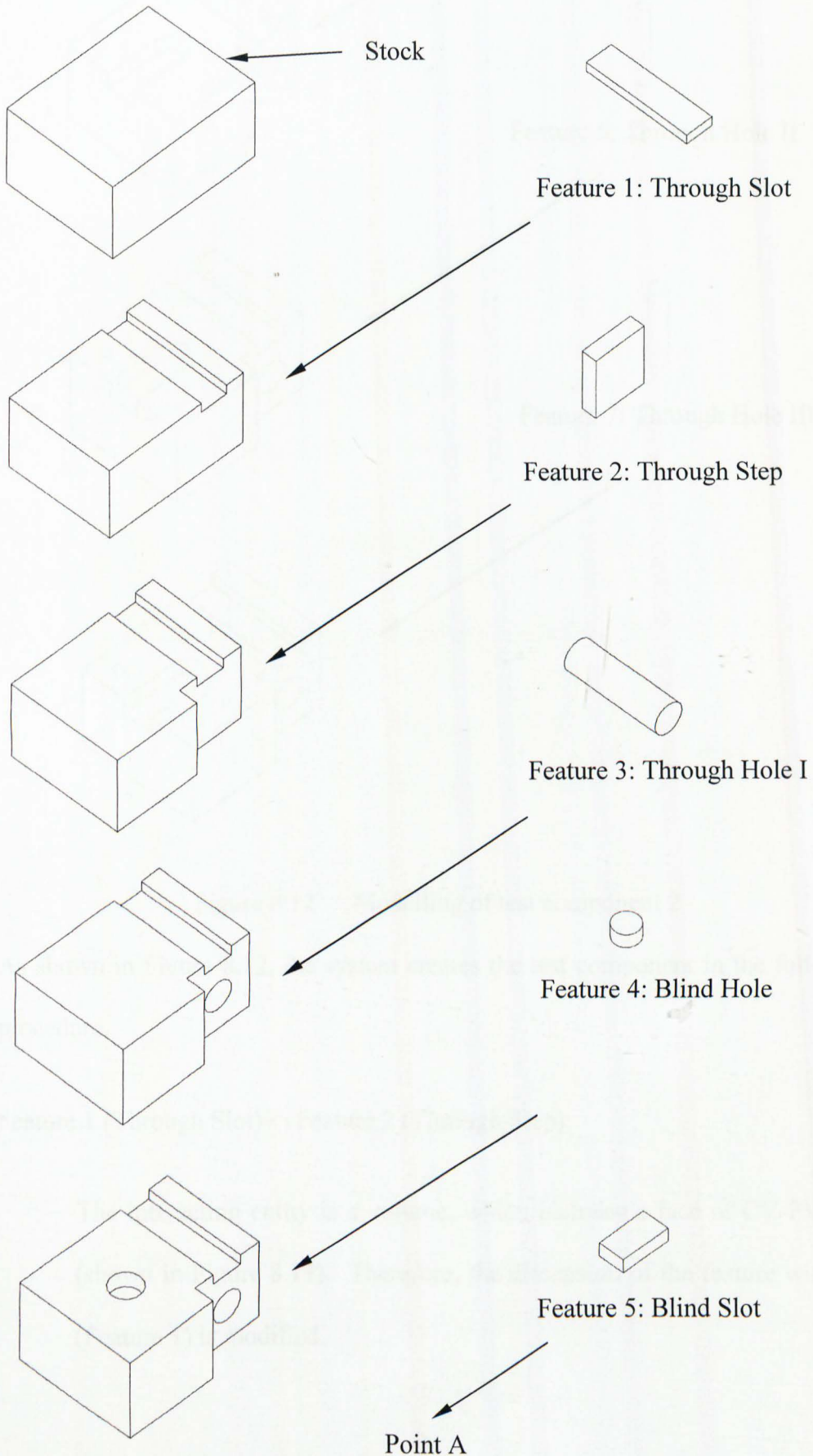
Groupno	flagno	featureno	processno	dx	dy	dz
1	1	1	1002	0	0	-1
2	1	3	1016	0	0	-1
3	1	2	1016	0	0	-1
4	1	5	1039	0	0	-1
5	1	4	1026	0	0	-1
6	0	6	1037	0	0	-1
		7	1037	0	0	-1

Figure 8.10 Machining operation groups test component 1

operationindex	groupno	setupno	setupname	featureno
1	1	10601	roughshaping	1
2	2	10111	rough slot milling	3
		20111	semirough slot milling	3
		30112	semifine slot milling	3
3	3	10111	rough slot milling	2
		20111	semirough slot milling	2
		30112	semifine slot milling	2
4	6	10201	drilling	6
		10201	drilling	7
		70401	reaming	7
		70401	reaming	6
5	4	10201	drilling	5
		10501	rough boring	5
		10401	rough reaming	5
		40401	finish reaming	5
6	5	10201	drilling	4
		10501	rough boring	4
		10401	rough reaming	4
		40401	finish reaming	4

Figure 8.11 Process plan generated for test component 1

8.4.2 Example 2



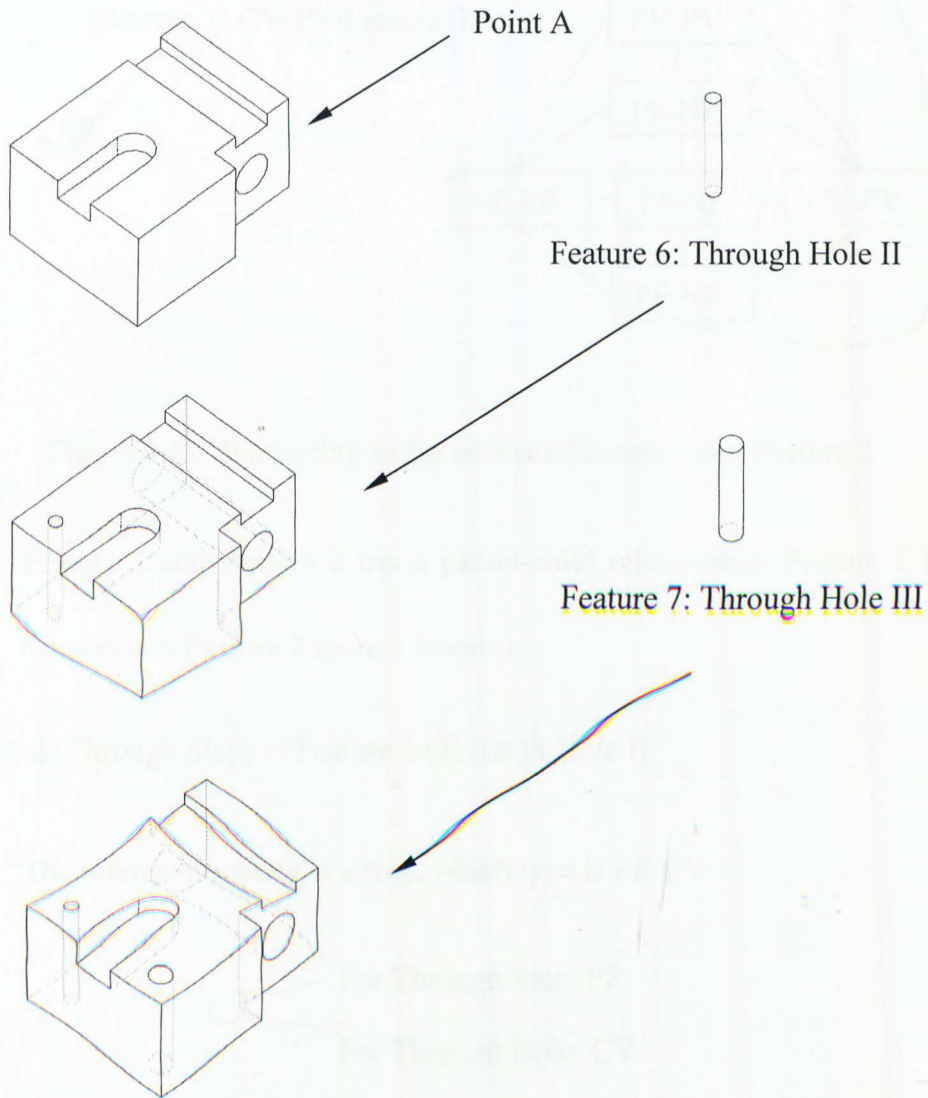


Figure 8.12 Modelling of test component 2

As shown in Figure 8.12, the system creates the test component in the following procedure.

Feature 1 (Through Slot) \cap Feature 2 (Through Step):

The interacting entity is a volume, which includes a face of CV-PV type (shown in Figure 8.13). Therefore, the dimension of the feature with CV (Feature 1) is modified.

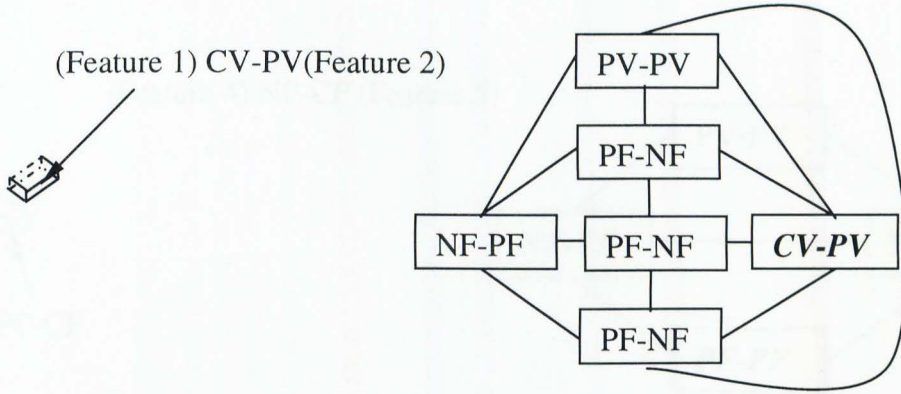
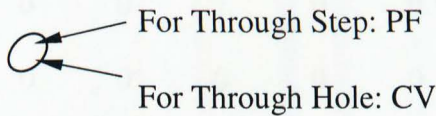


Figure 8.13 Interacting entity between Feature 1 and Feature 2

Feature 1 and Feature 2 has a parent-child relationship: Feature 1 (child feature) and Feature 2 (parent feature).

Feature 2 (Through Step) \cap Feature 3 (Through Hole I):

The interacting entity is a face, which type is PF-CV:



There is a parent-child relationship between Feature 3 (child feature) and Feature 3 (parent feature).

Feature 4 (Blind Hole) \cap Feature 5 (Blind Slot):

The interacting entity is a volume, which has a face of NF-CF and a face of PF-PF shown in Figure 8.14. Thus, Feature 4 and Feature 5 are merged into one feature: Blind Slot. The F-Adjacent vector and V-Adjacent vector of the merged feature are stored in files of neurlin.txt and neuravf.txt, respectively:

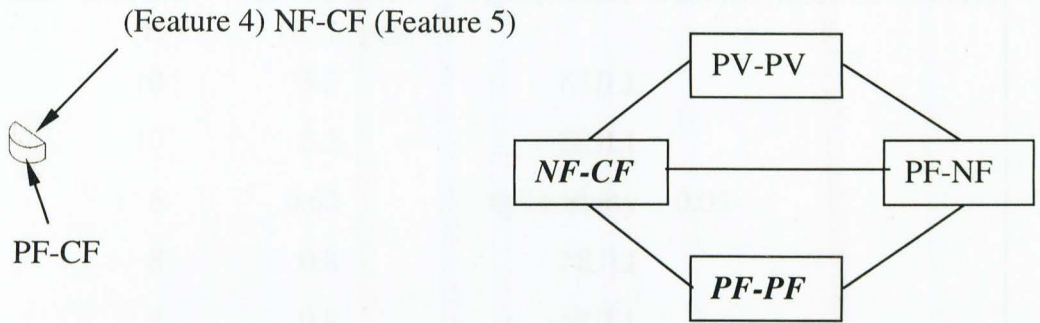


Figure 8.14 Interacting entity between Feature 4 and Feature 5

File neurlin.txt (F-Adjacent vector):

6	1	9	3	0	2	1	3	0	6
	3	0	6	0	0				

File neuravf.txt (V-Adjacent vector):

1	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	1
	0	0							

The tolerance and surface finish of the features are shown in Figure 8.15.

Figure 8.16 gives the feature recognition result of this example. It can be shown that the proposed methodology skips features for which no recognition is necessary, and is therefore more efficient. For example, the neural network-based feature recogniser is only executed for interacting features: Feature 4 and Feature 5, with satisfactory outcome of recognition.

featureno	tolerance (IT)	surface finish (R_a in μm)	shape/location tolerance(referencefeature)
1	10	3.2	NULL
2	10	3.2	NULL
3	6	0.63	cylindricity 0.05
4	8	0.8	NULL
5	8	0.8	NULL
6	8	0.63	NULL
7	8	0.63	NULL

Figure 8.15 Tolerance and surface finish of the features of test component 2

The feature precedence list, machining operation groups and process planning of this example are shown in Figure 8.17, Figure 8.18 and Figure 8.19. The result shows that the final process plan is appropriate, where Feature 2 is machined prior to Feature 1; Features 5 and 6 are grouped; and Features 1 and 4 are machined successively.

the 1 feature
 the original feature class is Though Slot
 the last feature class is Though Slot
 the dimension of feature is changed and become child feature of the 2 feature

the 2 feature
 the original feature class is Though Step
 the last feature class is Though Step
 the feature changed the dimension of the 1 feature and is parent feature of the 1 feature
 the feature is the parent feature of feature 3

the 3 feature
 the original feature class is Through Hole
 the last feature class is Through Hole
 the feature is the child feature of feature 2

the 4 feature
 the original feature class is Blind Hole
 the last feature class is Blind Hole
 the feature is merged with the 5 feature type is Blind Slot

the 5 feature
 the original feature class is Blind Slot
 the last feature class is Blind Slot
 the feature is merged with the 4 feature type is Blind Slot

the 6 feature
 the original feature class is Through Hole
 the last feature class is Through Hole

the 7 feature
 the original feature class is Through Hole
 the last feature class is Through Hole

Figure 8.16 Feature recognition result of test component 2

index	featureno	typeno	flagno
1	2	Through Step	1
2	1	Through Slot	1
3	3	Through Hole	1
4	(4,5 merge) 4	Blind Slot	0
5	(original 6) 5	Through Hole	0
6	(original 7) 6	Through Hole	0

Figure 8.17 Feature precedence list of test component 2

groupno	flagno	featureno	processno	dx	dy	dz
1	1	2	1005	0	1	0
2	1	1	1016	0	0	-1
3	1	3	1026	0	1	0
4	0	4	1056	0	0	-1
5	0	5	1024	0	0	-1
		6	1024	0	0	-1

Figure 8.18 Machining operation groups test component 2

operationindex	groupno	setupno	setupname	featureno
1		1		
		10131	rough step milling	2
		20131	semirough step milling	2
		30132	semifine step milling	2
2		2		
		10111	rough slot milling	1
		20111	semirough slot milling	1
		30112	semifine slot milling	1
3		4		
		10112	rough slot milling	4
		20112	semirough slot milling	4
		30114	semifine slot milling	4
4		5		
		10201	Drilling	5
		10201	Drilling	6
		70401	reaming	6
		70401	reaming	5
5		3		
		10201	Drilling	3
		10501	rough boring	3
		10401	rough reaming	3
		40401	finish reaming	3

Figure 8.19 Process planning for test component 2

8.4.3 Example 3

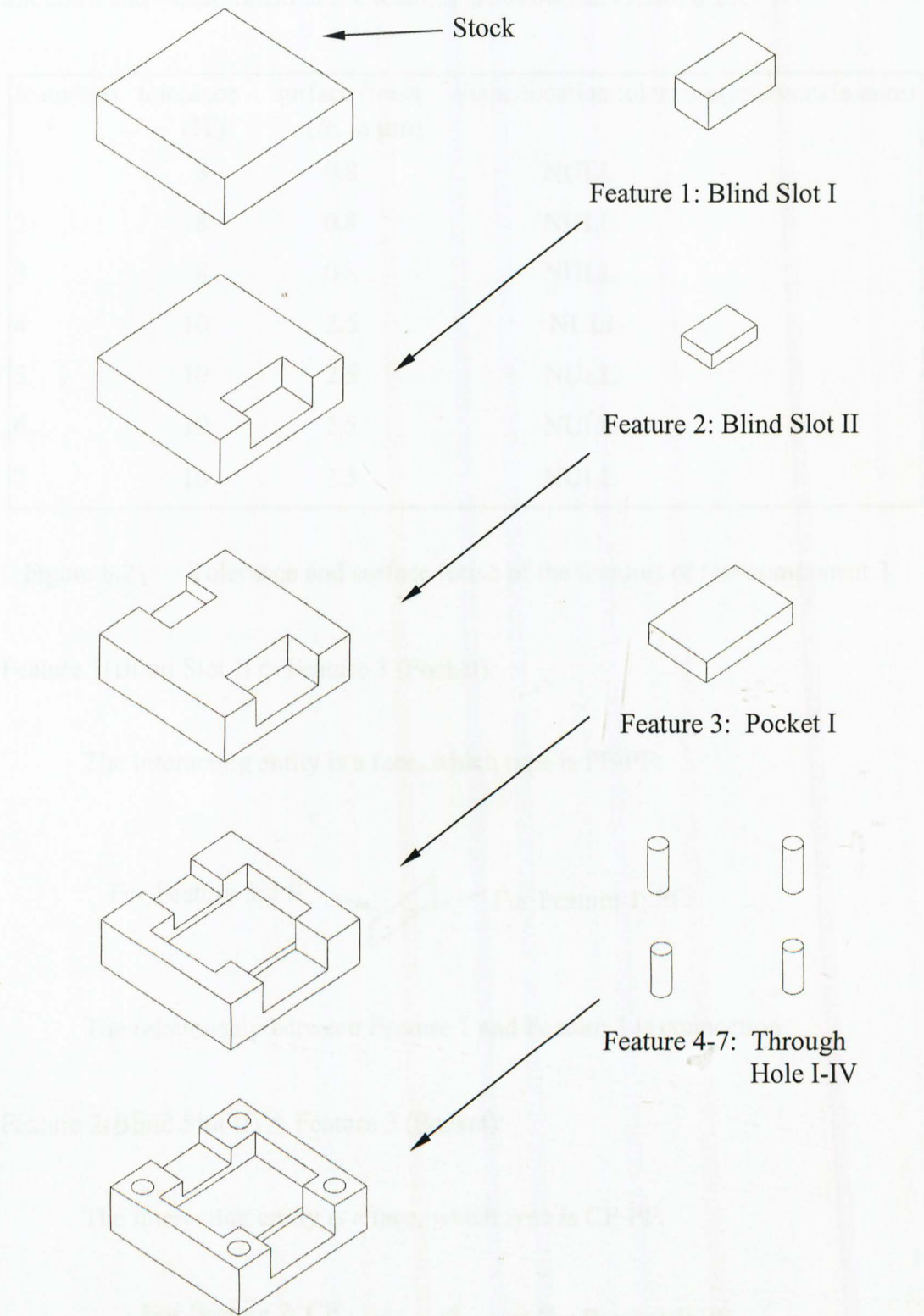


Figure 8.20 Modelling of test component 3

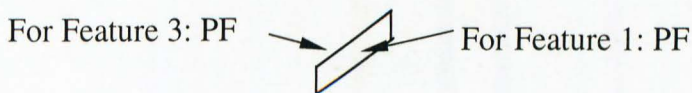
The test component has been designed in the procedure given in Figure 8.20. The tolerance and surface finish of the features are shown in Figure 8.21.

feature no	tolerance (IT)	surface finish (R_a in μm)	shape/location tolerance(reference feature)
1	8	0.8	NULL
2	8	0.8	NULL
3	8	0.8	NULL
4	10	2.5	NULL
5	10	2.5	NULL
6	10	2.5	NULL
7	10	2.5	NULL

Figure 8.21 Tolerance and surface finish of the features of test component 3

Feature 1(Blind Slot I) \cap Feature 3 (Pocket):

The interacting entity is a face, which type is PF-PF:



The relationship between Feature 1 and Feature 3 is connection.

Feature 2(Blind Slot II) \cap Feature 3 (Pocket):

The interacting entity is a face, which type is CF-PF:



The class of Feature 2 is changed from Blind Slot to Through Slot. There is a parent-child relationship between Feature 2 (child feature) and Feature 3 (parent feature).

Figure 8.22 shows the feature recognition result of this example. From this example, it can be found that the proposed method can differentiate between the two interacting situations efficiently: Features 1 and 3, and Features 2 and 3.

The feature precedence list, machining operation groups and process planning of this example are given in Figure 8.23, Figure 8.24 and Figure 8.25. It can be seen that the results are good, e.g. Features 4 to 7 form a group because they have the same class and tool approach directions, accuracy requirements and can be manufactured by the same processes.

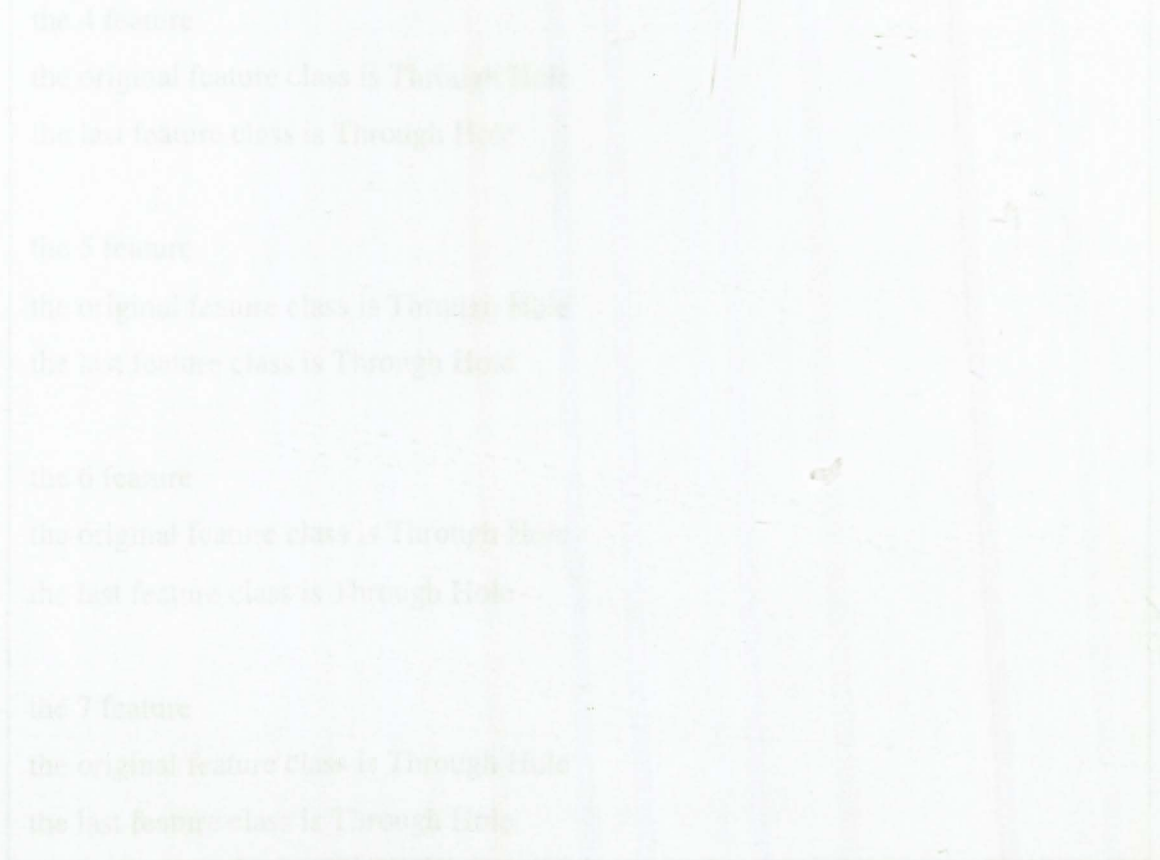


Figure 8.22 Feature recognition result of test component 3

<p>the 1 feature</p> <p>the original feature class is Blind Slot</p> <p>the last feature class is Blind Slot</p> <p>the feature has a connect relationship with the feature 3</p>
<p>the 2 feature</p> <p>the original feature class is Blind Slot</p> <p>the last feature class is Through Slot</p> <p>the feature has been changed feature class and become child feature of the 3 feature</p>
<p>the 3 feature</p> <p>the original feature class is Pocket</p> <p>the last feature class is Pocket</p> <p>the feature has a connect relationship with the feature 1</p> <p>the feature has changed the 2 feature's class and is parent feature of the 2 feature</p>
<p>the 4 feature</p> <p>the original feature class is Through Hole</p> <p>the last feature class is Through Hole</p>
<p>the 5 feature</p> <p>the original feature class is Through Hole</p> <p>the last feature class is Through Hole</p>
<p>the 6 feature</p> <p>the original feature class is Through Hole</p> <p>the last feature class is Through Hole</p>
<p>the 7 feature</p> <p>the original feature class is Through Hole</p> <p>the last feature class is Through Hole</p>

Figure 8.22 Feature recognition result of test component 3

index	featureno	typeno	flagno
1	1	Blind Slot	0
2	3	Closed Pocket	1
3	2	Through Slot	1
4	4	Through Hole	0
5	5	Through Hole	0
6	6	Through Hole	0
7	7	Through Hole	0

Figure 8.23 Feature precedence list of test component 3

groupno	flagno	featureno	processno	dx	dy	dz
1	0	1	1056	0	0	-1
2	1	3	1067	0	0	-1
3	1	2	1016	0	0	-1
4	0	4	1024	0	0	-1
		5	1024	0	0	-1
		6	1024	0	0	-1
		7	1024	0	0	-1

Figure 8.24 Machining operation groups test component 3

operationindex	groupno	setupno	setupname	featureno
1	4	10201	Drilling	5
		10201	Drilling	6
		10201	Drilling	4
		10201	Drilling	7
		70401	reaming	7
		70401	reaming	4
		70401	reaming	5
		70401	reaming	6
2	2	10121	rough pocket milling	3
		20121	semirough pocket milling	3
		30121	semifine pocket milling	3
3	3	10111	rough slot milling	2
		20111	semirough slot milling	2
		30112	semifine slot milling	2
4	1	10112	rough slot milling	1
		20112	semirough slot milling	1
		30114	semifine slot milling	1

Figure 8.25 Process planning of test component 3

8.5 Summary

This chapter has described the experimental implementation of a CAD/CAPP prototype system with typical examples to demonstrate system capabilities. The implementation has shown genuine integration of CAD and CAPP under the proposed methodology using various contemporary techniques.

Chapter 9.

Conclusions

To summarise the research presented in this thesis, this chapter draws conclusions and proposes recommendations for further work.

9.1 Research contributions

This thesis has detailed the research on integrated CAD/CAM through CAPP using feature technology. An experimental implementation of a prototype system has been carried out for prismatic components. A number of test components have been presented to demonstrate the capabilities of the methodology.

Main activities considered include

- 1) Component design based on a number of standard feature classes with validity check.
- 2) Search of interacting features and identification of features relationships.
- 3) Recognition of new features formed by interacting features.
- 4) Production of a feature based model for the component.
- 5) Generation of a suitable process plan covering selection of machining operations, grouping of machining operations and process sequencing.

In order to carry out above activities, a number of algorithms and methods have been proposed.

1) Machining feature extraction

The existing methods did not solve problems of the interacting features efficiently and concisely. The research has led to a more efficient and simpler solution. The proposed method has the following highlights:

- A new feature classification for machining application. The main characteristic is the adoption of STEP AP224 and multi-viewpoint of design and manufacture.
- Feature-based model management dealing with adding, editing and deleting features. Constraints for feature validity are checked to effectively maintain the model validity for the component in terms of geometry and topology.
- A novel heuristic algorithm to recognise interacting features. The Interacting Entity between each feature pair is analysed instead of the new volume created by all interacting features used in conventional approaches, simplifying the process. In addition, the algorithm skips features for which no recognition is necessary, and is therefore more efficient. All interacting entities between feature pairs can be detected, reported and handled in an appropriate way. Invalid operations that cause constraint violation of the model validity are tackled effectively.
- Neural network-based techniques for feature recognition to improve the capability of current methods. The proposed input representation, F-adjacent

matrix and V-adjacent matrix, not only solves the problems of ambiguity and overlaps successfully, but also describes the parallel relationships which previous work cannot provide. The conjugate gradient algorithm trains the net in the directions with the fastest convergence. The hierarchical structure for feature classifiers is more suitable for feature classification. The successful results for classification at both the first and secondary levels demonstrate the ability to generate the required cluster from the proposed input representation.

- A unified data structure for feature class in the feature-based model. New feature classes can be defined using the same data structure.

2) Feature-based CAPP

CAPP requires a suitable representation of the component to be manufactured. A method for CAPP has been proposed making use of the features extracted from the feature based component model.

- A suitable process planning database containing six libraries. A management module has been developed with an open and flexible structure, making the CAPP system adaptive to dynamic environments.
- A precedence algorithm to identify feature precedence relations based on the precedence constraints. Simplification has been made by considering the design intention.

- A strategy for optimal process sequencing. The most important characteristic is the multi-objective fitness function designed to consider minimum manufacturing cost, shortest manufacturing time and the best satisfaction of process sequence rules, which previous research does not consider simultaneously.
- The application of the analytical hierarchical process (AHP) to evaluating the satisfaction degree of process sequence rules for process sequencing. AHP not only transfers the manufacturing sequence constraint problems into a numerical solution, but also makes the evaluation more adaptive.
- A method to allocate the relative weights for the three main evaluating factors for process sequencing, using intelligent neural network and fuzzy technique. The method can adapt the relative weights for the evaluating factors according to various component, customer and production requirements, offering greater flexibility.

9.2 Limitations

The methodology presented in the thesis, however, has certain limitations, which are described below.

- 1) Restricted component geometry, i.e. only 3D prismatic components are considered.

- 2) Limited standard feature classes. Currently, the work deals only with internal features.
- 3) Lack of consideration of fixturing.
- 4) Lack of optimisation of the machining parameters for individual operations.

9.3 Recommendations for future work

This thesis has proposed a methodology for CAD/CAM integration based on feature technology and artificial intelligence techniques. It is apparent from the results and limitations of the research that further work is necessary.

- 1) Further work on neural network-based feature recognition

As demonstrated in Chapter 5, neural network-based feature recognition has been successful to recognise and classify feature at both the first and second levels. The features can be further classified at the third level according to CAPP requirements for further optimisation.

- 2) Extension of the domain of component geometry

Although the proposed method considers the majority of features - internal features which are likely to be of interest for the application of process planning, external features and attaching features should be covered in the future.

- 3) Consideration of fixturing

Fixturing may affect dramatically process planning. Future effort should be paid to fixturing constraints and corresponding clamping strategy.

4) Interfacing with NC systems

The research can be extended to interface with NC systems to cover toolpath generation and planning for a complete process plan.

In conclusion, the research carried out will help to resolve feature interactions and to further automate process planning, thus achieving genuine integration of design and manufacturing.

References

Anderson D.C. and Chang T.C., “Automated process planning using objected-oriented feature-based design”, *Advanced Geometric Modelling for Engineering Applications*, Elsevier Science Publishers B.V., pp. 247, 1990

Bhaskara Reddy S.V., Shunmugam M.S. and Narendran T.T., “Operation sequencing in CAPP using genetic algorithms”, *International Journal of Production Research*, Vol. 37, No. 5, pp. 1063-1074, 1999

Bhandarker P.M, Nagi R., “STEP-based feature extraction from STEP geometry for Agile Manufacturing”, *Computers in Industry*, Vol. 41, pp. 3-24, 2000

Bidarra R. and Bronsvoort W.F., “Semantic feature modelling”, *Computer-Aided Design*, Vol. 32, No. 2, pp. 201-225, 2000

Bronsvoort W.F. and Jansen F.W., “Feature modelling and conversion - key concepts to concurrent engineering”, *Computers in Industry*, Vol. 21, pp. 61-86, 1993.

Case K. and Harun W.A.W, “A single representation to support assembly and process planning in feature-based design machined parts”, *Proceedings of the Institution of Mechanical Engineerings Part B-Journal of Engineering Manufacture*, Vol. 213, No. 2, pp. 143-155, 1999

Case K. and Hounsell M.S., “Feature modelling: a validation methodology and its evaluation”, *Journal of Materials Processing Technology*, Vol. 107, pp. 15-23, 2000

- Cay F. and Constantin C.**, “An IT view on perspectives of computer aided process planning research”, *Computers in Industry*, Vol. 34, pp. 307-337, 1997
- Chan, C.C.H.**, “ANN-based feature recognition and grammar-based feature extraction to integrate design and manufacturing”, *PhD Thesis*, University of Iowa, USA, 1994.
- Chan, K.C. and Nhieu, J.**, “A Framework for Feature-based Applications”, *Computers and Industrial Engineering*, Vol. 24, No. 2, pp. 151-164, 1993.
- Chang T.C.**, *Expert Process Planning for Manufacturing*, Addison-Wesley, New York, 1989
- Charalambous, C.**, “Conjugate gradient algorithm for efficient training of artificial neural networks”, *IEE Proceedings –G Circuits Devices and Systems*, Vol. 139, No. 3, pp. 301-310, 1992
- Chen, C.L.P. and LeClair, S.R.**, “Unsupervised neural learning algorithm for setup generation in process planning”, *Proceedings of International Conference on Artificial Neural Networks in Engineering*, pp. 663-668, 1993.
- Chen, Y.H. and Lee, H.M.**, “A neural network system for 2D feature recognition”, *International Journal of Computer Integrated Manufacturing*, Vol. 11, No. 2, pp. 111-117, 1998.
- Choi B.K., Barash M.M. and Anderson D.C.**, “Automatic recognition of machined surfaces from a 3D solid model”, *Computer Aided Design*, Vol.16, No. 2, pp. 81-86, 1984

References

Chu C.C.P., and Gadh R., "Feature-based approach for set-up minimisation of process design from product design", *Computer-Aided Design*, Vol. 28, No. 5, pp. 321-332, 1996

Chuang, J.H., Wang, P.H. and Wu, M.C., "Automatic classification of block-shaped parts based on their 2D projections", *Computers & Industrial Engineering*, Vol. 36, No. 3, pp. 697-718, 1999.

Chung, J.C.H., Patel D.R., Cook R.L. and Simmons M.K., "Feature-based modeling for mechanical design", *Computer & Graphics*, Vol. 14, No. 2, pp. 189-199, 1990.

Dagli, C.H., Poshyanonda, P. and Bahrami, A., "Neuro-computing and concurrent engineering", In Parsaei, HR and Sullivan, WG (eds), *Concurrent Engineering: Contemporary Issues and Modern Design Tools*, pp.465-486, Chapman & Hall, London, 1993.

De Floriani L. and Bruzzone E., "Building a feature-based object description from a boundary model", *Computer Aided Design*, Vol. 21, No. 10, pp. 602-610, 1989

De Martino T., Falcidieno B., Giannini F., Hassinger S. and Ovtcharova J., "Feature-based modelling by integrating design and recognition approaches", *Computer-aided Design*, Vol. 26, No. 8, pp. 646-653, 1994

Demuth H and Beale M, *Neural Network Toolbox For Use with MATLAB*, The MATH WORKS Inc., 2000

Dereli T. and Filiz H., “Optimisation of process planning functions by genetic algorithms”, *Computers & Industrial Engineering*, Vol. 36, pp. 281-308, 1999

Dereli T., Filiz H. and Baykasoglu A., “Optimizing cutting parameters in process planning of prismatic parts by genetic algorithms”, *International Journal of Production Research*, Vol. 39, No. 15, pp. 3303-3328, 2001

Desai V.S. and Pande S.S., “GFM—an interactive feature modeller for CAPP of rotational components”, *Computer-Aided Engineering Journal*, Vol. 8, pp. 217, 1991

Devireddy, C.R. and Ghosh, K., “Feature-based modelling and neural networks-based CAPP for integrated manufacturing”, *International Journal of Computer Integrated Manufacturing*, Vol. 12, No. 1, pp. 61-74, 1999.

Ding L., Yue Y. and Ahmet K., “An integrated approach to integrating CAD/CAM”, *Proceedings of 6th Annual Conference of the Chinese Automation and computer Society in the UK*, pp. 51-64, Loughborough, 23-24 September 2000

Dixon J.R., Libardi E.C., Luby S.C., Valghul M., and Simmones M.K., “Examples of Symbolic Representations of Design Geometric Engineering with Computers “, *Expert Systems for Mechanical Design*, pp. 1-10, 1987.

Donaldson I.A. and Corney J.R., “Rule-based feature recognition for 2.5D machined components”, *International Journal of Computer Integrated Manufacturing*, Vol. 6, No. 1-2, pp. 51-64, 1993

Dong, J.X., Tang, X.Q. and Wang, S.C., “Inference mechanism for CAPP tool

References

based on artificial neural network”, *Proceedings of 1st Congress on Intelligent Manufacturing*, Puerto Rico, pp. 994-1002, 1995.

Fausett, L., *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, Prentice Hall International, Inc, 1994.

Gindy, N.N.Z., “A hierarchical structure for form features”, *International Journal of Production Research*, Vol 27, No 12, pp 2089-2103, 1989.

Gindy, N.N.Z., Yue, Y. and Zhu, C.F., “Automated feature validation for creating / editing feature-based component data models”, *International Journal of Production Research*, Vol 36, No. 9, pp. 2479-2495, 1998.

Giusti, F., Santochi, M. and Dini, G., "COATS: an expert module for optimal tool selection", *Annals of the CIRP*, Vol. 35, No. 1, pp. 337-340, 1986.

Golden, B.L., Harker, P.T. and Wasil, E.E., *The Analytic Hierarchy Process: Applications and Studies*, Springer-Verlay, Berlin, 1989

Goslin J., *Xgenetic: an ActiveX genetic algorithm control*, www.regnow.com, 2000

Gu, Z., Zhang, Y.F. and Nee, A.Y.C., “Generic form feature recognition and operation selection using connectionist modelling”, *Journal of Intelligent Manufacturing*, Vol. 6, No. 4, pp. 263-273, 1995.

Gu, Z., Zhang, Y.F. and Nee, A.Y.C., “Identification of important features for machining operations sequence generation”, *International Journal of Product Research*, Vol. 35, No. 8, pp. 2285-2307, 1997.

Gurney, K., *An Introduction to Neural Networks*, UCL Press, 1997

References

- Han J.H., Kang M. and Choi H.**, "STEP-based feature recognition for manufacturing cost optimization", *Computer-Aided Design*, Vol. 33, pp. 671-686, 2001
- Hounsell M.S. and Case K.**, "Feature-based interaction: an identification and classification methodology", *Proc Instn Mech Engrs, Part B*, Vol.213, pp. 369-380, 1999
- Hwang, J.L.**, "Applying the perceptron to 3D feature recognition", *PhD Thesis*, Arizona State University, USA, 1991.
- Jasthi, S.R.K., Prasad A.V.S.R.K., Manidhar G., Rao P.N., Rao U.R.K. and Tewari N.K.**, "A feature-based part description system for computer-aided process planning", *Journal of Design and Manufacturing*, Vol. 4, pp. 67, 1994.
- Joshi, S.B. and Chang, T.C.**, "Graph-based heuristics for recognition of machined features from a 3D solid model", *Computer Aided Design*, Vol. 20, No. 2, pp. 58-66, 1988.
- Kang T.S. and Nnaji B.O.**, "Feature Representation and Classification for Automatic Process Planning Systems", *Journal of Manufacturing Systems*, Vol. 12, No. 2, pp. 133-145, 1993
- Kim Y.S.**, "Recognition of form features using convex decomposition", *Computer Aided Design*, Vol. 24, No. 9, pp. 461-476, 1992
- Kim K. and Jeong J.**, "Finding feasible tool-approach directions for sculptured surface manufacture", *IIE Transactions*, Vol. 28, No. 10, pp. 829-836, 1996
- Knapp, G.D. and Wang, H.P.**, "Acquiring, storing and utilising process planning

References

knowledge using neural networks”, *Journal of Intelligent Manufacturing*, Vol. 3, No. 5, pp. 333-344, 1992.

Laakko T. and Mantyla M., “A new form feature recognition algorithm”, *Computer Applications in Production and Engineering: Integration Aspects*, pp. 369-376, 1991

Lam S.M and Wong T.N., “Recognition of machining features – a hybrid approach”, *International Journal of Production Research*, Vol. 38, No. 17, pp. 4301-4316, 2000

Lee J.Y. and Kim K., “A feature-based approach to extracting machining features”, *Computer Aided Design*, Vol. 30, No. 13, pp. 1019-1035, 1998

Lee J.Y. and Kim K., “Generating alternative interpretations of machining features”, *International Journal of Advanced Manufacturing Technology*, Vol. 15, pp. 38-48, 1999

Le Tumelin, C., Garro, O. and Charpentier, P., “Generating process plans using neural networks”, *Proceedings of 2nd International Workshop on Learning in Intelligent Manufacturing Systems*, Budapest, Hungary, 1995.

Lentz D.H. and Sowerby R., “Feature extraction of concave and convex regions and their intersections”, *Computer Aided Design*, Vol. 25, No. 7, pp. 421-437, 1993

Li, R.K., Tu, Y.M. and Yang T.H., “Composite feature and variational design concepts in a feature-based design system”, *International Journal of Production Research*, Vol. 31, No. 7, pp. 1521-1540, 1993

References

Li, Y., Mills, B., Moruzzi, J.L. and Rowe, W.B., “Grinding wheel selection using a neural network”, *Proceedings of the 10th National Manufacturing Research Conference*, Loughborough, pp. 597-601, 1994.

Li W.D., ONG S.K. and Nee A.Y., “Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts”, *Production Research*, Vol. 40, No. 8, pp. 1899-1922, 2002

Mei, J., Zhang, H.C. and Oldham, W.J.B., “A neural network approach for datum selection in computer-aided process planning”, *Computers in Industry*, Vol. 27, No. 1, pp. 53-64, 1995.

Miao, H.K., Sridharan, N. and Shah, J.J., “CAD-CAM integration using machining features”, *International Journal of Computer Integrated Manufacturing*, Vol. 15, No. 4, pp. 296-318, 2002.

Nezis, K. and Vosniakos, G., “Recognising 2.5D shape features using a neural network and heuristics”, *Computer Aided Design*, Vol 29, No. 7, pp. 523-439, 1997.

Osakada, K. and Yang, G.B., “Neural networks for process planning of cold forging”, *Annals of the CIRP*, Vol. 40, No. 1, pp. 243-246, 1991.

Öztürk N. and Öztürk F., “Neural network based non-standard feature recognition to integrate CAD and CAM”, *Computers in Industry*, Vol. 45, pp. 123-135, 2001.

References

- Park, M.W., Rho, H.M. and Park, B.T.**, “Generation of modified cutting condition using neural network for an operation planning system”, *Annals of the CIRP*, Vol. 45, No. 1, pp. 475-478, 1996.
- Park, M.W., Park, B.T., Rho, Y.M. and Kim, S.K.**, “Incremental supervised learning of cutting conditions using the Fuzzy ARTMAP neural network”, *Annals of the CIRP*, Vol 49, No. 1, pp. 375-378, 2000.
- Patterson D.W**, *Artificial neural networks*, London, Prentice-Hall, 1996.
- Perng, D.B. and Chang C.F.**, “A New Feature-based design system with Dynamic Editing”, *Computers and Industrial Engineering*, Vol. 32, No. 2, pp. 383-397, 1997.
- Peters, T.J.**, “Encoding mechanical design features for recognition via neural nets”, *Research in Engineering Design*, Vol. 4, No. 2, pp. 67-74, 1992.
- Pratt, M.J.**, “A hybrid feature-based modelling systems”, *Advanced Geometric Modelling for Engineering Applications*, Elsevier Science Publishers B.V., pp. 189, 1990
- Prabhakar, S. and Henderson, M.R.**, “Automatic form-feature recognition using neural-network-based techniques on B-rep of solid models”, *Computer Aided Design*, Vol. 24, No. 7, pp. 381-393, 1992
- Principe, J.C., Euliano, N.R. and Lefebvre, W.C.**, *Neural and adaptive system: Fundamentals through Simulations*, John Wiley & Sons, Inc, 2000.
- Qiao L., Wang X.Y, and Wang S.C.**, “A GA-based approach to machining operation sequencing for prismatic parts”, *International Journal of Production*

References

Research, Vol. 38, No. 14, pp. 3282-3303, 2000

Roy, R., Chodnikiewicz, K. and Balendra, R., "Interpolation of forging preform using neural networks", *Journal of Materials Processing Technology*, Vol. 45, Nos. 1-4, pp. 695-702, 1994.

Saaty T. L., *The Analytic Hierarchy Process*, McGraw hill, New York, 1980.

Saaty T. L., *Multicriteria Decision Making: The Analytic Hierarchy Process*, RWS Publications, Pittsburgh, PA, 1990.

Saaty T. L., *Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process*, RWS Publications, Pittsburgh, PA, 1994.

Sakakura, M. and Inasaki, I., "A neural network approach to the Decision-making process for grinding operations", *Annals of the CIRP*, Vol. 41, No. 1, pp. 353-356, 1992.

Santochi, M. and Dini, G., "Use of neural networks in automated selection of technological parameters of cutting tools", *Computer Integrated Manufacturing Systems*, Vol. 9, No. 3, pp. 137-148, 1996.

Sarma, S.E. and Wright, P.K., "Algorithms for the minimization of setups and tool changes in "simple fixturable" components in milling", *Journal of Manufacturing Systems*, Vol. 15, No. 2, pp. 95-112, 1996.

Shah J.J. and Rogers M.T., "Expert form feature modelling shell", *Computer Aided Design*, Vol. 20, pp. 515-524, 1988

Shah, J.J, and Mäntylä M., *Parametric and Feature-based CAD/CAM*, John Wiley & Sons, Inc., 1995

References

- Shan, X.H., Nee, A.Y.C. and Poo, A.N.**, “Integrated application of expert systems and neural networks for machining operation sequencing”, *Neural Networks in Manufacturing and Robotics*, ASME, PED-vol 57, pp 117-126, 1992.
- Shunmugan, M.S., Bhaskara, S.V. and Narendran, T.T.**, “Selection of optimal conditions in multi-pass face-milling using a genetic algorithm”, *International Journal of Machine Tools & Manufacture*, vol 40, pp 401-414, 2000
- Sreevalsan P. and Shah J.J.**, “Unification of feature definition methods”, *Intelligent Computer Aided Design*, Elsevier Science Publishers, pp. 83-100, 1992
- Srinath, G.**, “Optimising neural net input for feature recognition”, *MSc Thesis*, Arizona State University, USA, 1993.
- STEP**, STandard for External representation of Product data, ISO 10303-224 (Industrial automation systems and integration. Product data representation and exchange: mechanical product definition for process plans using machining features), 1999
- Taha H.A.**, *Operations Research--an Introduction*, Maxwell Macmillan International, 1997
- Tang K. and Woo T.C.**, “Algorithmic aspects of alternating sum of volumes”, *Computer Aided Design*, Vol. 23, No. 5-6, 1991
- Tseng Y.J. and Joshi S.B.**, “Recognition of interacting rotational and prismatic machining features from 3-D mill-turn parts”, *International Journal of Production Research*, Vol. 36, No. 11, pp. 3147-3165, 1998

References

- Tseng Y.J.**, "A modular modeling approach by integrating feature recognition and feature-based design", *Computers in Industry*, Vol. 39, pp. 113-125, 1999
- Turner G.P. and Aderson D.C.**, "An object-oriented approach to interactive, feature-based design for quick turnaround manufacturing", *Computers in Engineering Conference*, ASME, San Francisco, July/Aug, pp. 551-555, 1988
- Vancza J. and Markus A.**, "Genetic Algorithms in process planning", *Computers in Industry*, Vol. 17, pp. 181-194, 1991
- Vandenbrande, J. and Requicha, A.**, "Spatial reasoning for automatic recognition of interacting form features", *Proceedings of the ASME computers in Engineering Conference*, Boston, Vol. 1, pp. 251-256, 1990
- Wong T.N. and Lam S.M.**, "Automatic recognition of machining features from computer aided design part models", *Proceedings of the Institution of Mechanical Engineerings Part B-Journal of Engineering Manufacture*, Vol. 214, pp. 515-520, 2000
- Woo T.C.**, "Feature extraction by volume decomposition", *Conference on CAD/CAM Technology in Mechanical Engineering*, MIT, Cambridge, MA, USA, pp. 76-94, 1982
- Woo Y. and Sakurai H.**, "Recognition of maximal features by volume decomposition", *Computer-Aided Design*, Vol. 34, pp. 195-207, 2002
- Wu, M.C. and Jen, S.R.**, "A neural network approach to the classification of 3D prismatic parts", *International Journal of Advanced Manufacturing Technology*, Vol. 11, No. 5, pp. 325-335, 1996.

Wu, J.J, Zhang T.B., Zhang X.F. and Zhou J., “A face based mechanism for naming, recording and retrieving topological entities”, *Computer Aided Design*, Vol. 33, pp. 687-698, 2001.

Yang W., Ding H. and Xiong Y., “Manufacturability analysis for a sculptured surface using visibility cone computation”, *The International Journal of Advanced Manufacturing Technology*, Vol. 15, pp. 317-321, 1999.

Yip-Hoi D. and Dutta D., “A genetic algorithm application for sequencing operations in process planning for parallel machining”, *IIE Transactions*, Vol. 28, No. 1, pp. 55-68, 1996.

Yue C.F. and Venuvinod Patri K., “Geometric feature recognition: coping with the complexity and infinite variety of features”, *International Journal of Computer Integrated Manufacturing*, Vol. 12, No. 5, pp. 439-452, 1999.

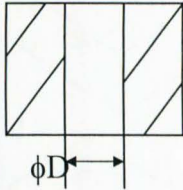
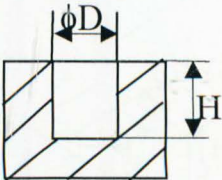
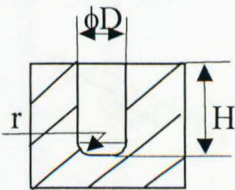
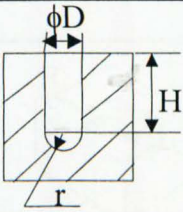
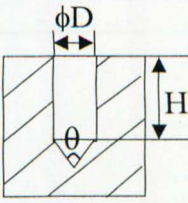
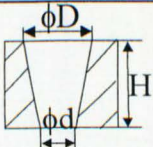
Zhang, C., Chan K.W. and Chen Y.H., “A method for recognising feature interactions and feature components within the interactions”, *The International Journal of Advanced Manufacturing Technology*, Vol. 13, pp. 713-722, 1997.

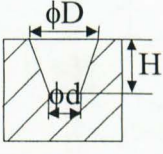
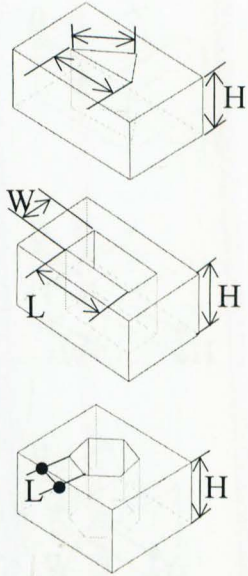
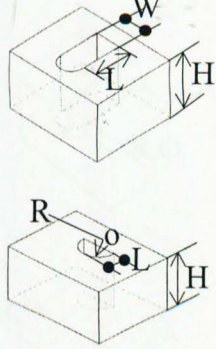
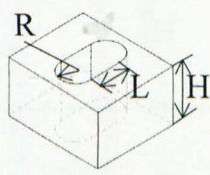
Zhao, F.L. and Wu P.S.Y., “A cooperative framework for process planning”, *International Journal of Computer Integrated Manufacturing*, Vol. 12, No. 2, pp. 168-178, 1999

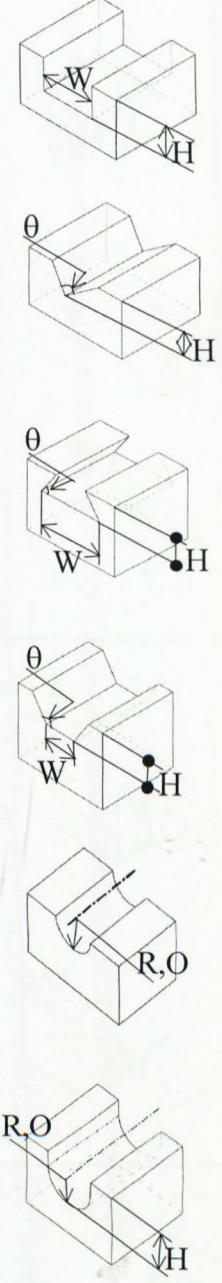
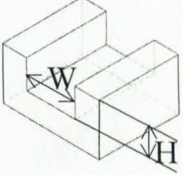
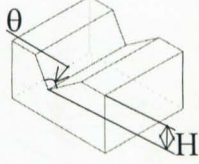
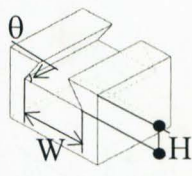
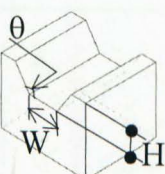
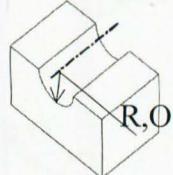
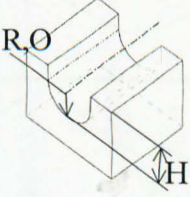
Zulkifli, A.H. and Meeran, S., “Feature patterns in recognising non-interacting and interacting primitive, circular and slanting features using a neural network”, *International Journal of Production Research*, Vol. 37, No. 13, pp. 3063-3100, 1999.

Appendix A

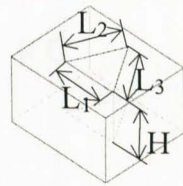
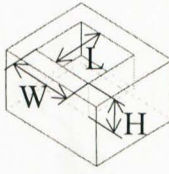
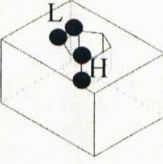
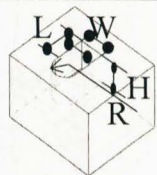
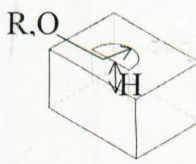
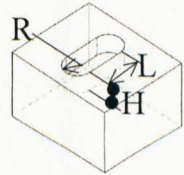
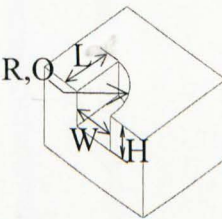
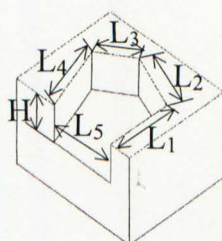
Feature Classification

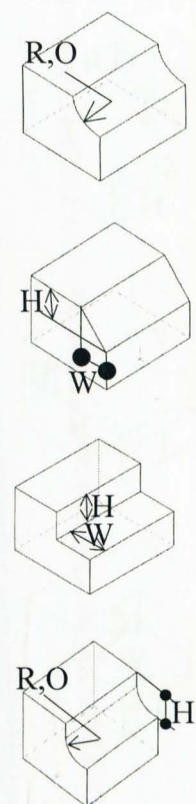
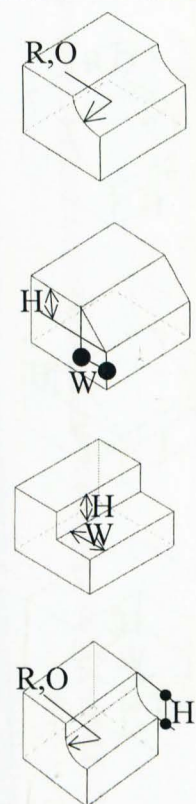
Internal feature				
<p>Internal features are those geometric entities which form the internal structure of a part and can be created with restricted tool and tool accessibility directions.</p>				
Round hole.	Drilling	Through round hole		
	Plunge-milling			
	Reaming			
	Boring			
		Blind round hole	Flat	
			Flat with radius	
			Spherical	
			Conical	
Conical hole	Taper boring	Through		

	Reaming Internal grinding	Blind		
General hole	Broach Internal shaping	Polygon hole		
	Broading Milling Internal Shaping	Part-circle hole		
	End Milling	Double-semi-circle hole		

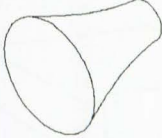
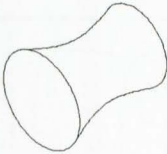
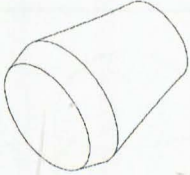
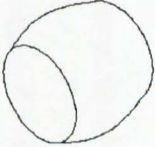
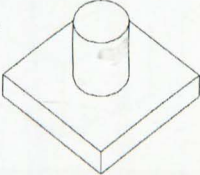
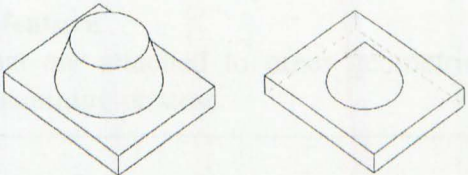
Slot	Milling	Through slot		
				
				
				
				
				
				

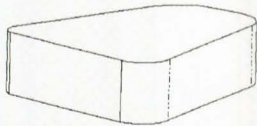
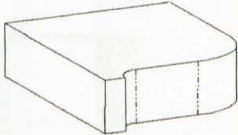
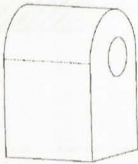

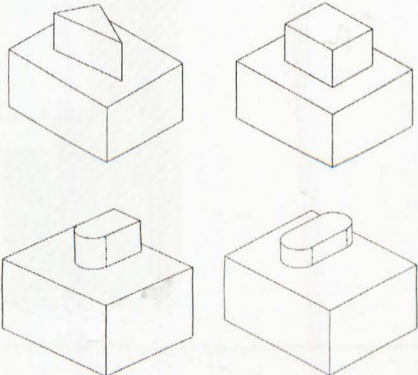
		Blind slot	Flat	
			Radiused	
			Woodraff	

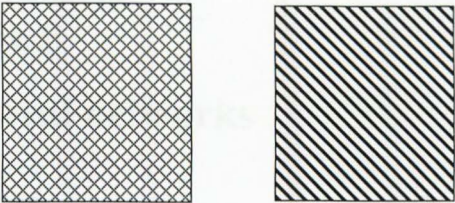
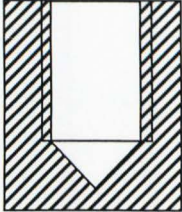

Pocket	End milling	Closed pocket	Polygon closed pocket	  
			Part-circle closed pocket	 
			Double-semi-circle closed pocket	
Milling	Open pocket		 	

Step	Milling/ Shaping (Planing)	Through step		
				

		Blind step		<p>The diagrams illustrate the geometry of a blind step in a rectangular block. The first diagram shows a top-down view of the step with labels R,O and H. The second diagram shows a side view with labels H, W, and L. The third diagram shows a top-down view of the step with labels H, W, and L. The fourth diagram shows a top-down view of the step with labels R,O and H. The fifth diagram shows a side view of the step with labels H, L₂, W₂, W₁, and L₂. The sixth diagram shows a side view of the step with labels H, R,O, L, and W.</p>
--	--	------------	--	--

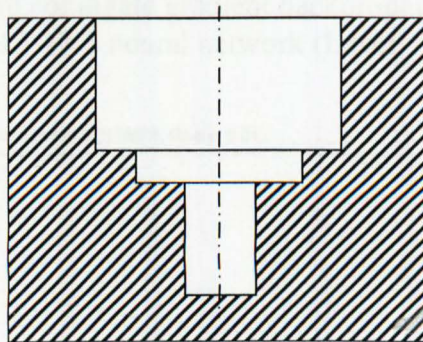
External feature			
Revolved- feature	Contour turning Profile forming turning	General revolution	
		Groove	
		Revolved_ flat	
		Revolved_ round	
Boss	Turning Grinding	Cylinder boss	
		Conical boss	

General outside	Milling	Close	
		Open	
Round_end	Milling		
Spherical cap	Profile-forming turning Lapping		
Protrusion	Milling		
Surface machining	Milling		
<p>Attaching feature Attaching features are special shapes that are attached to other geometrical features and can be created by a special machining process.</p>			

Knurl surface	Knurling		
Thread	External threading Internal threading Threading milling Tapping		
Marking: text on a metal surface			

Compound feature

Compound features are certain complex entities which are created by a special union of two or more machining features.



Appendix B

Experiments of neural networks

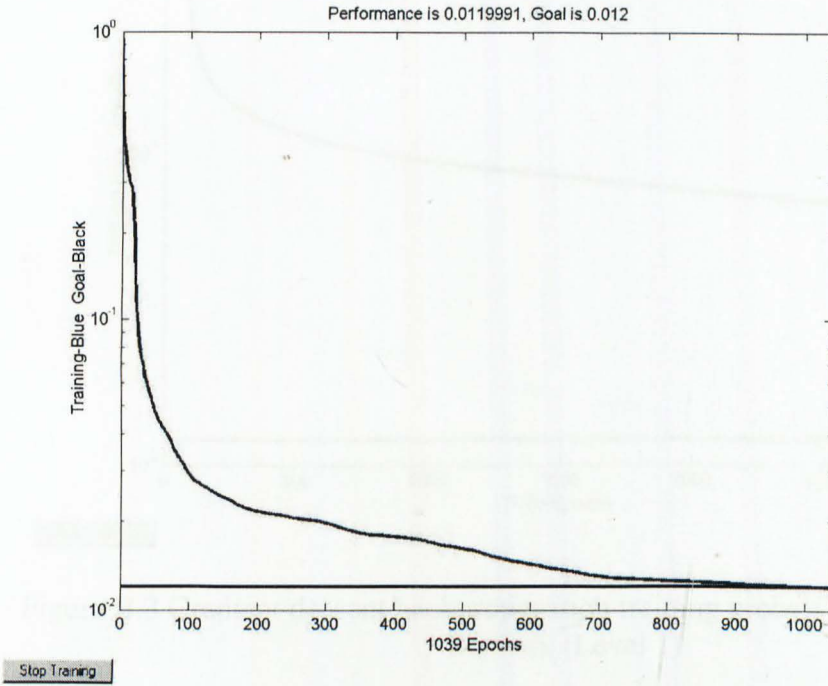


Figure B.1 The Polak-Ribiere conjugate gradient backpropagation training process of 15-17-5 neural network (Level 1)

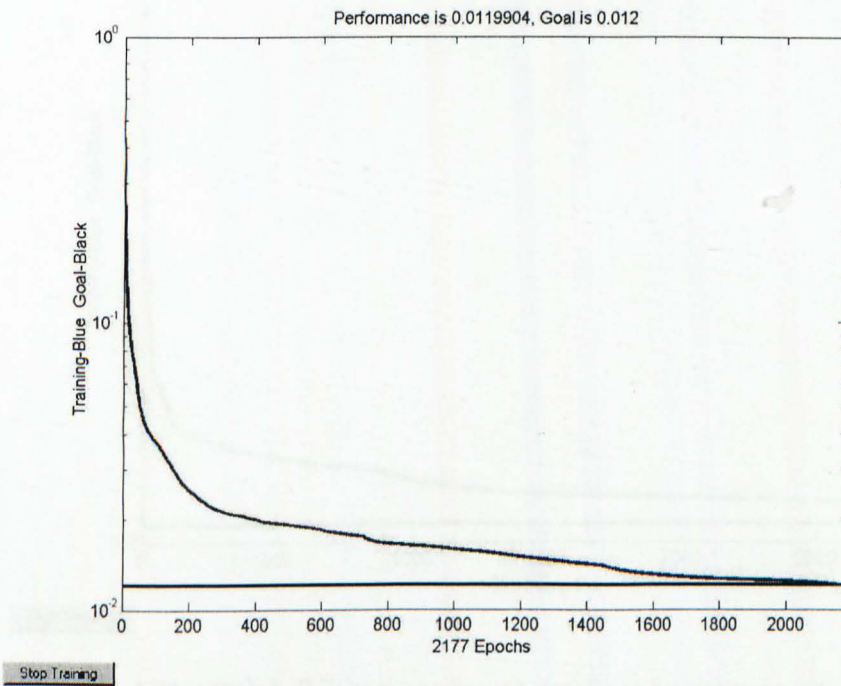


Figure B.2 The Fletcher-Powell conjugate gradient backpropagation training process of 15-17-5 neural network (Level 1)

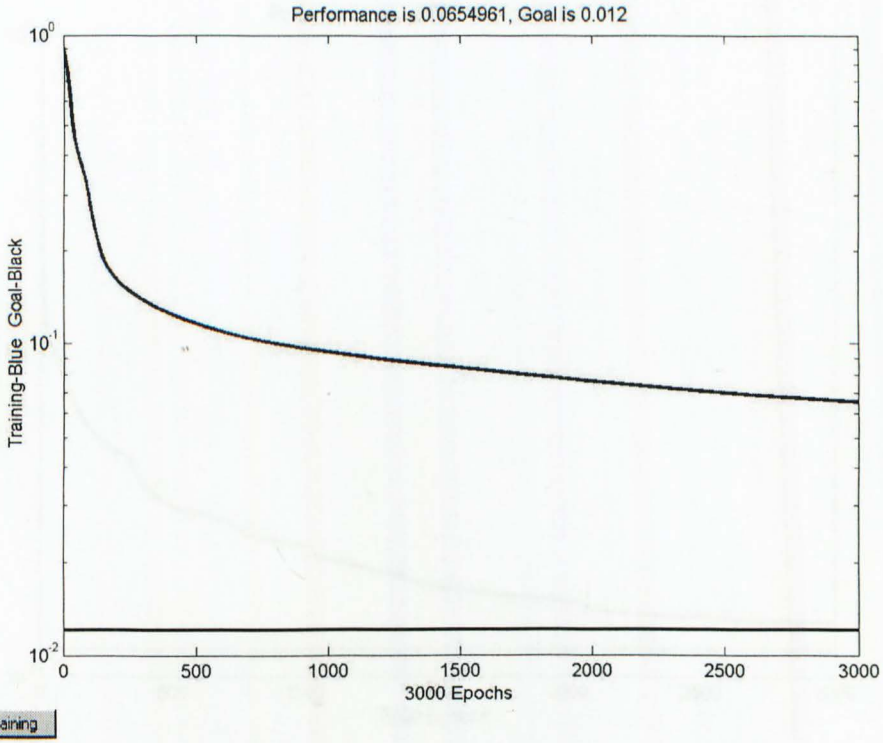


Figure B.3 Gradient descent backpropagation training process of 15-17-5 neural network (Level 1)

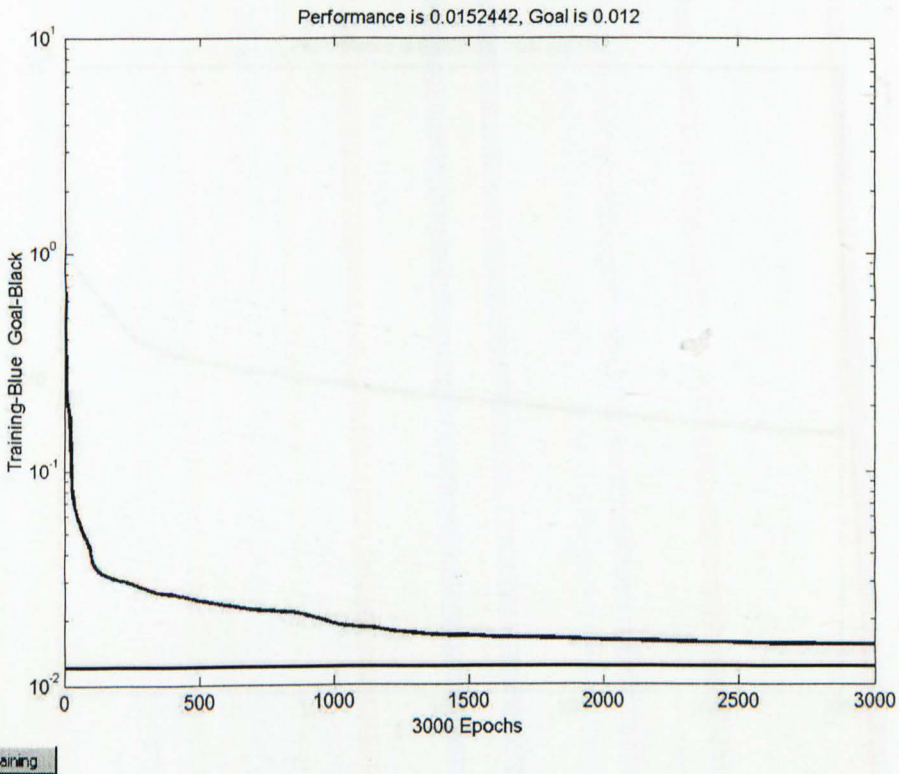


Figure B.4 The Polak-Ribiere conjugate gradient backpropagation training process of 15-16-5 neural network (Level 1)

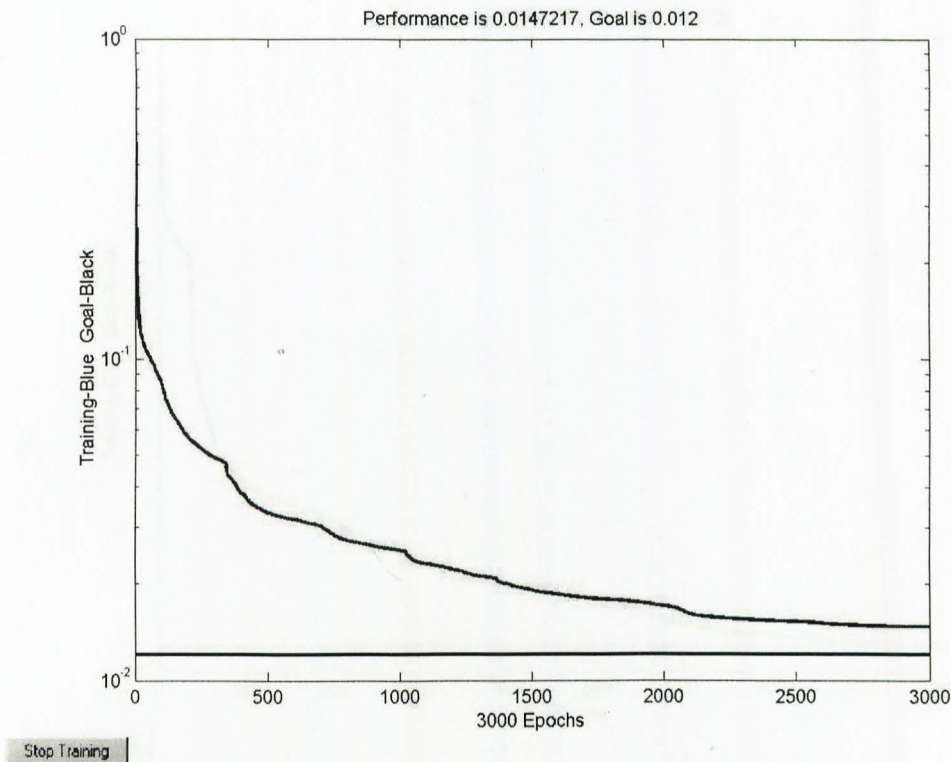


Figure B.5 The Fletcher-Powell conjugate gradient backpropagation training process of 15-16-5 neural network (Level 1)

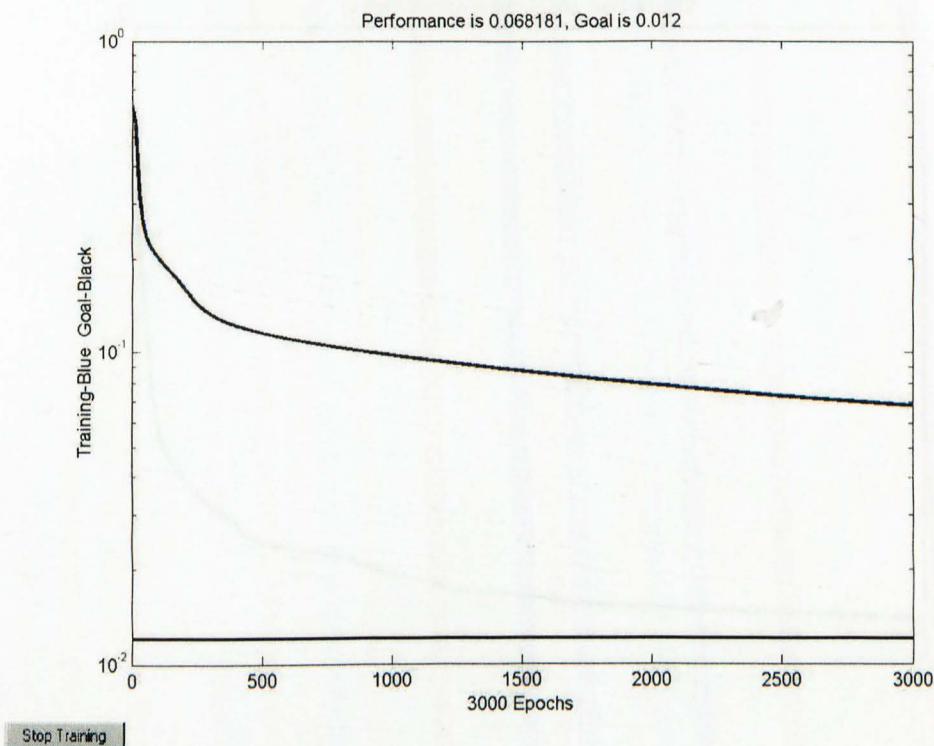


Figure B.6 Gradient descent backpropagation training process of 15-16-5 neural network (Level 1)

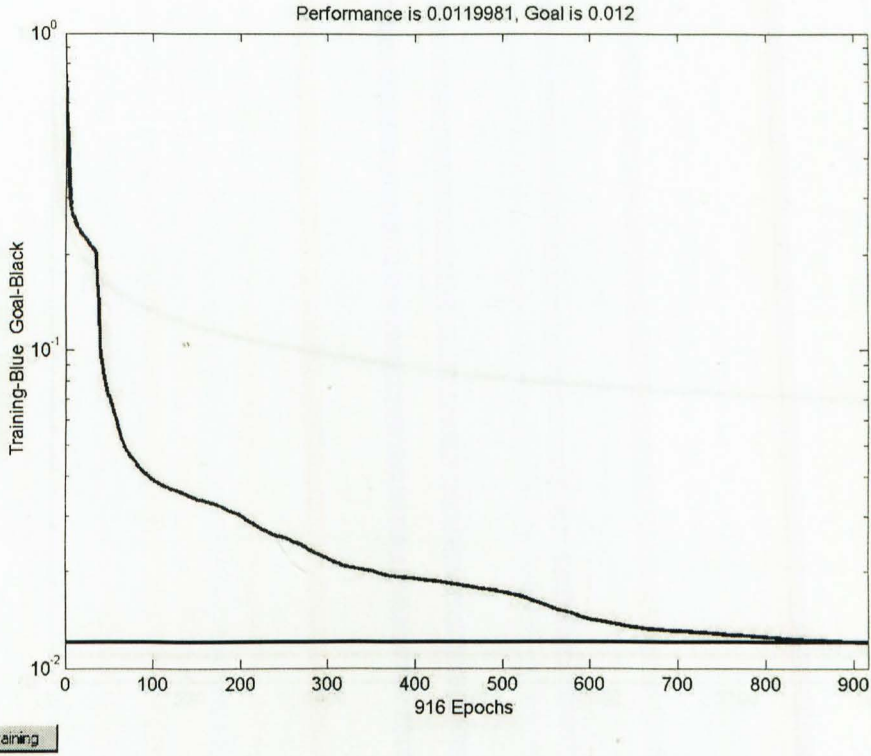


Figure B.7 The Polak-Ribiere conjugate gradient backpropagation training process of 15-18-5 neural network (Level 1)

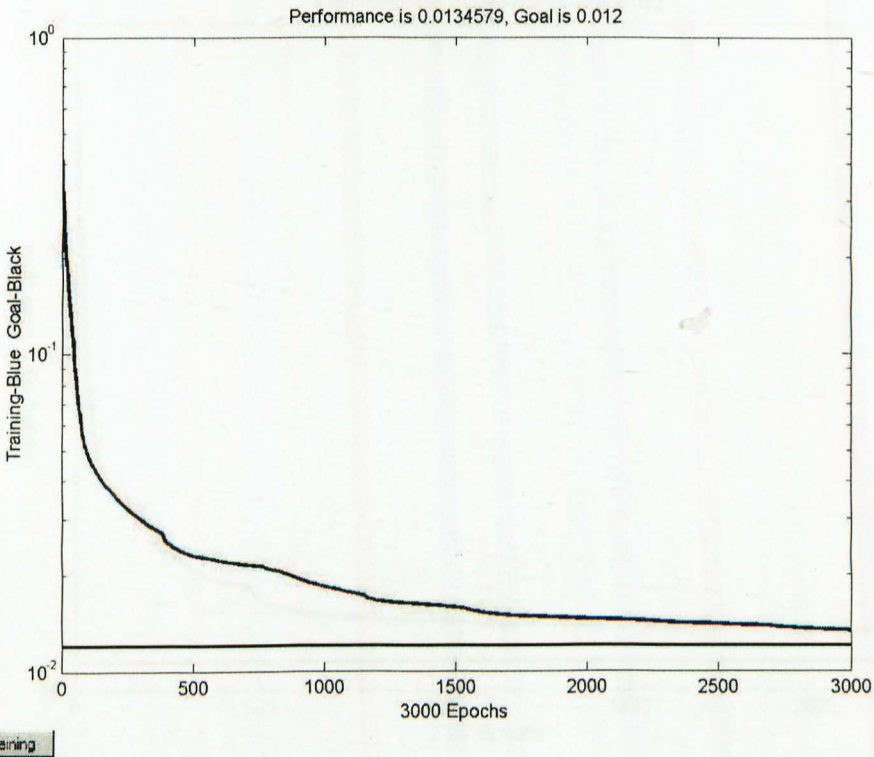


Figure B.8 The Fletcher-Powell conjugate gradient backpropagation training process of 15-18-5 neural network (Level 1)

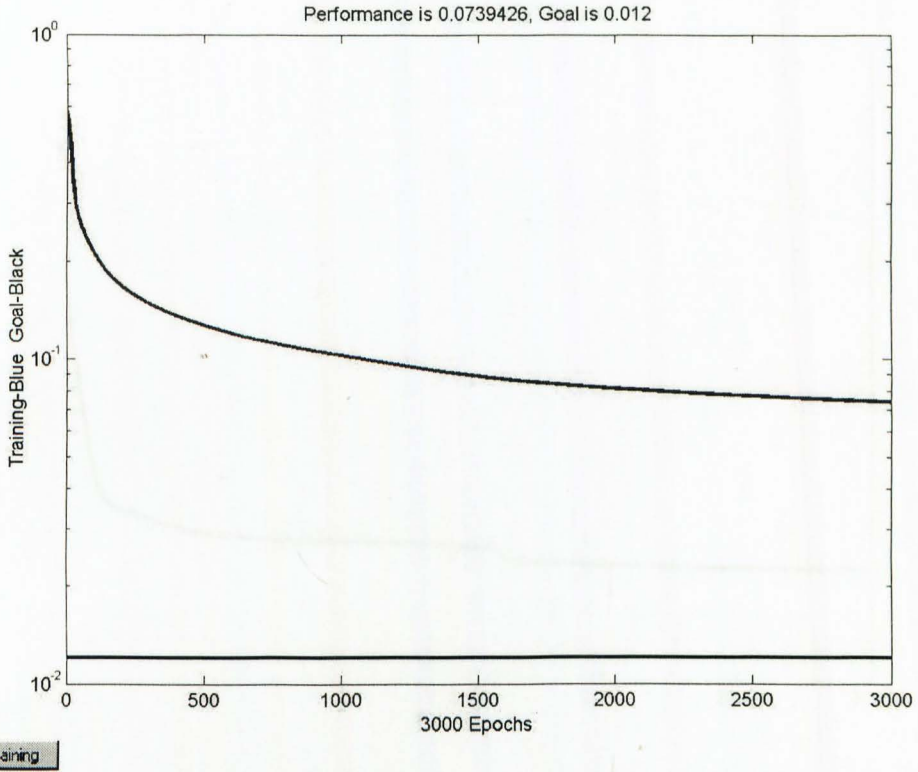


Figure B.9 Gradient descent backpropagation training process of 15-18-5 neural network (Level 1)

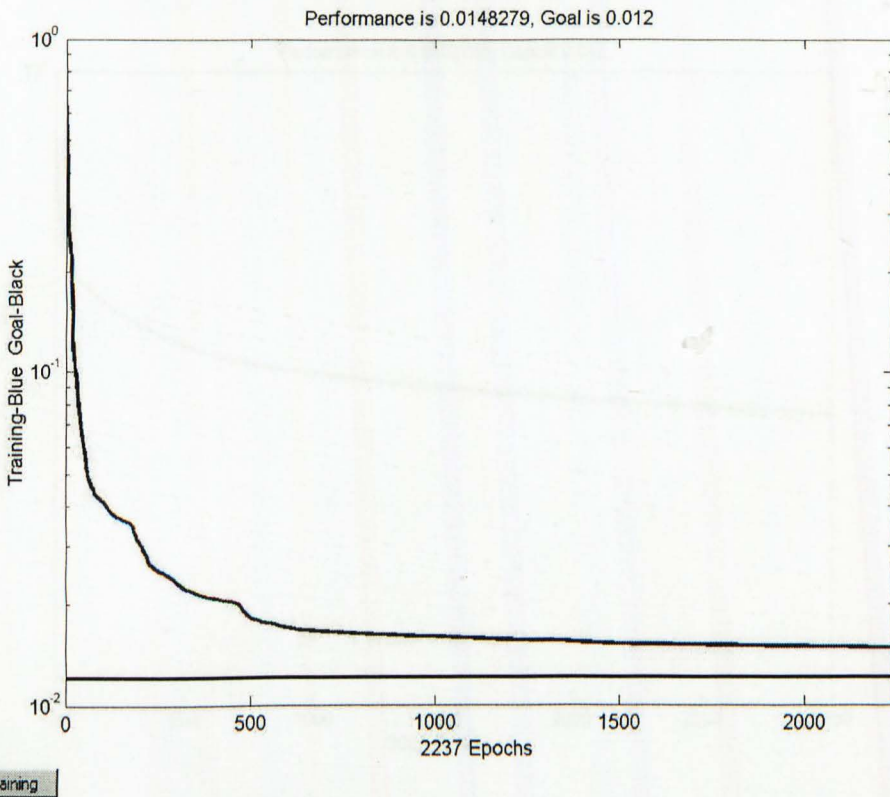


Figure B.10 The Polak-Ribiere conjugate gradient backpropagation training process of 15-15-5 neural network (Level 1)

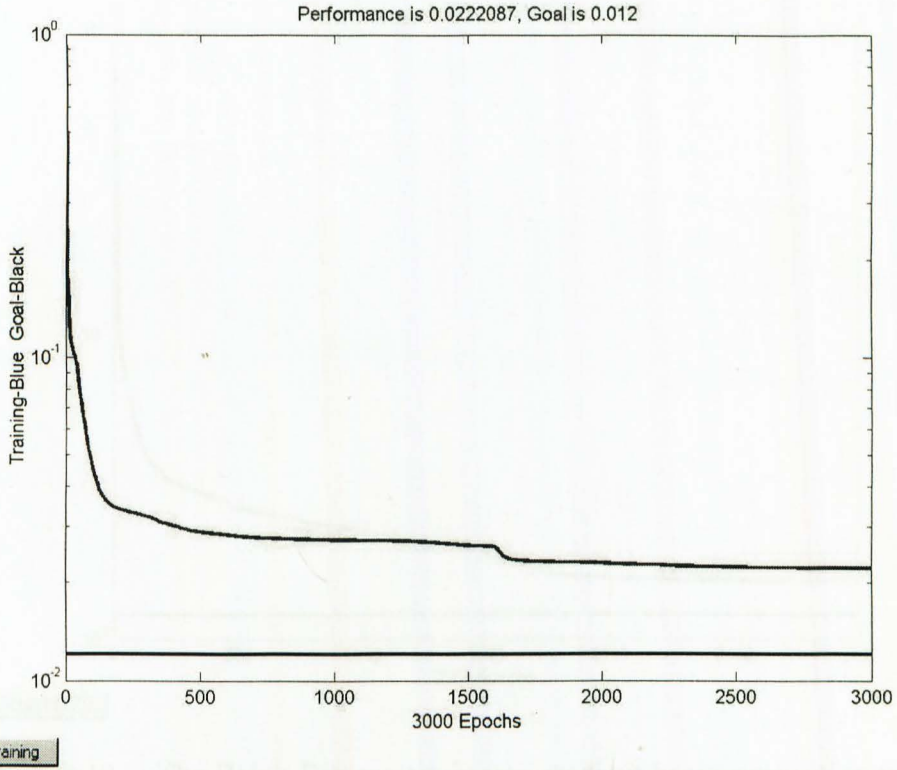


Figure B.11 The Fletcher-Powell conjugate gradient backpropagation training process of 15-15-5 neural network (Level 1)

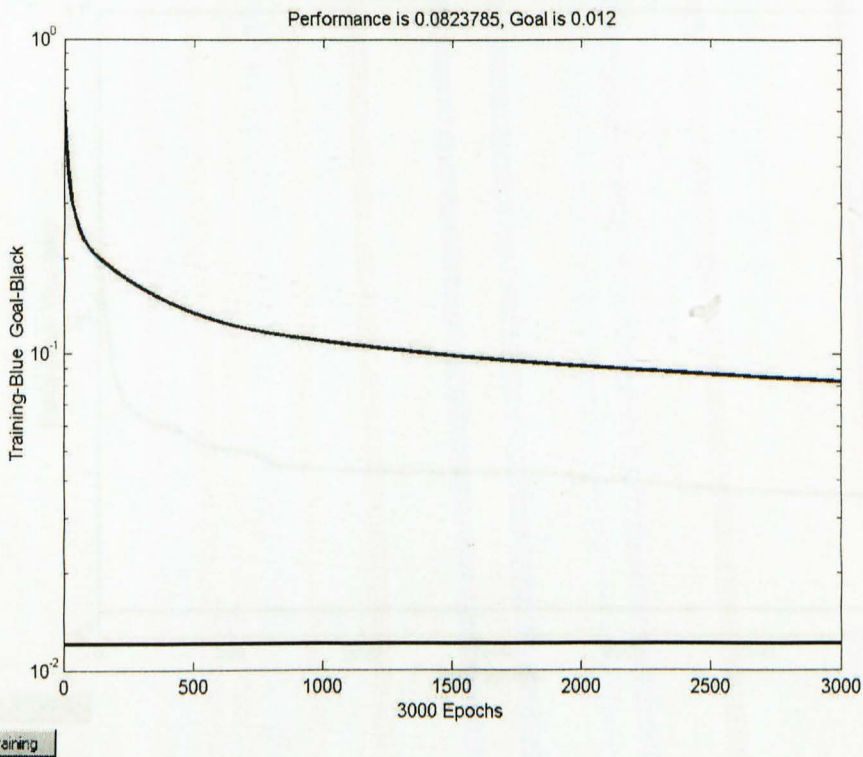


Figure B.12 Gradient descent backpropagation training process of 15-15-5 neural network (Level 1)

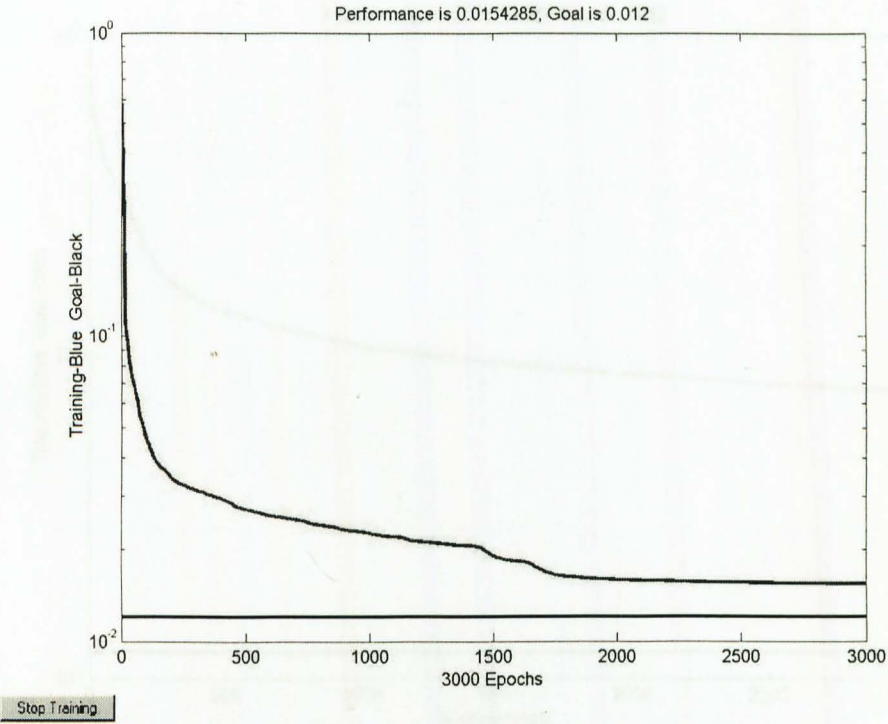


Figure B.13 The Polak-Ribiere conjugate gradient backpropagation training process of 15-14-5 neural network (Level 1)

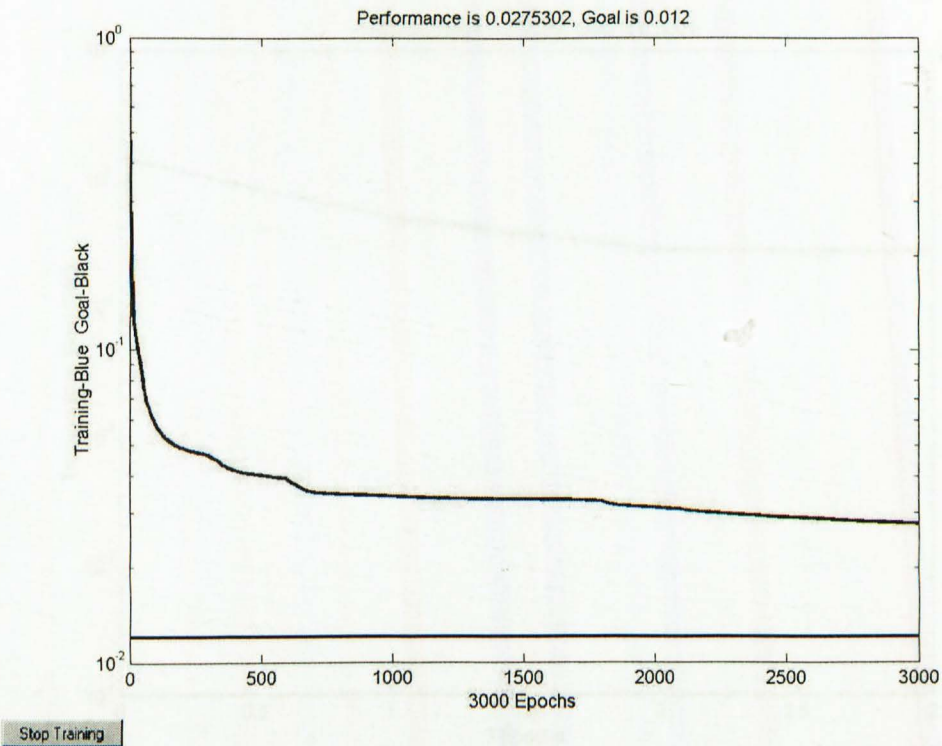


Figure B.14 The Fletcher-Powell conjugate gradient backpropagation training process of 15-14-5 neural network (Level 1)

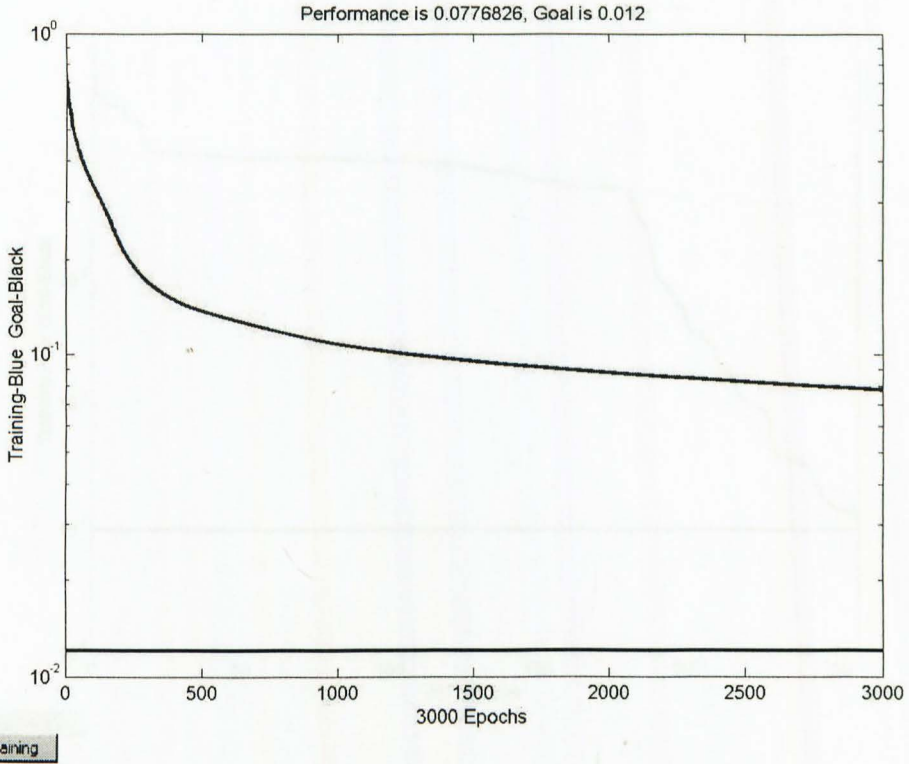


Figure B.15 Gradient descent backpropagation training process of 15-17-5 neural network (Level 1)

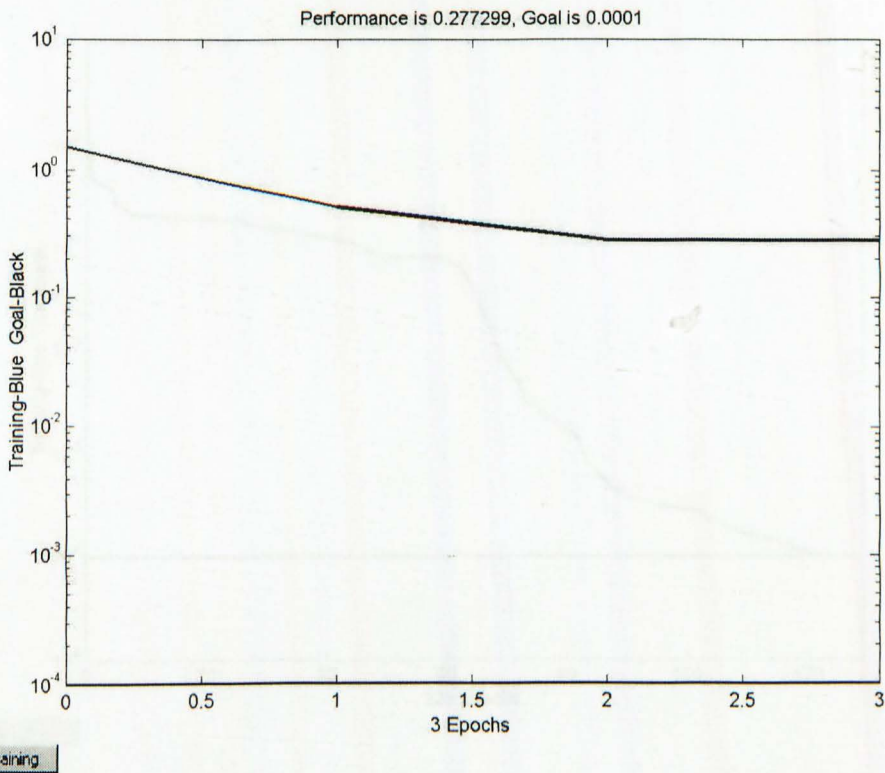


Figure B.16 The Polak-Ribiere conjugate gradient backpropagation training process of 21-1-2 neural network (Round Hole)

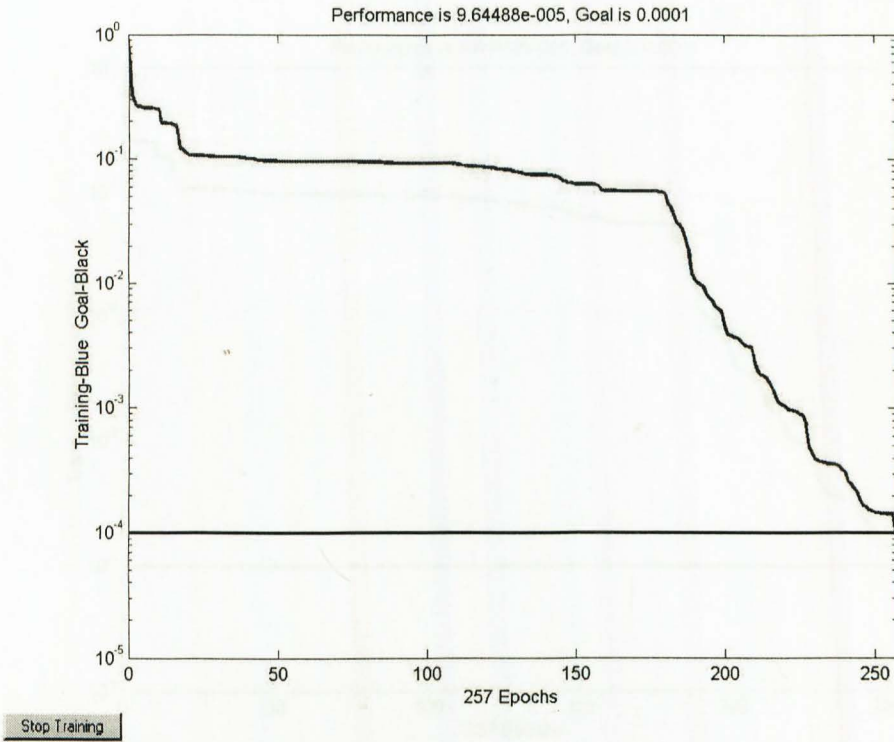


Figure B.17 The Polak-Ribiere conjugate gradient backpropagation training process of 21-2-2 neural network (Round Hole)

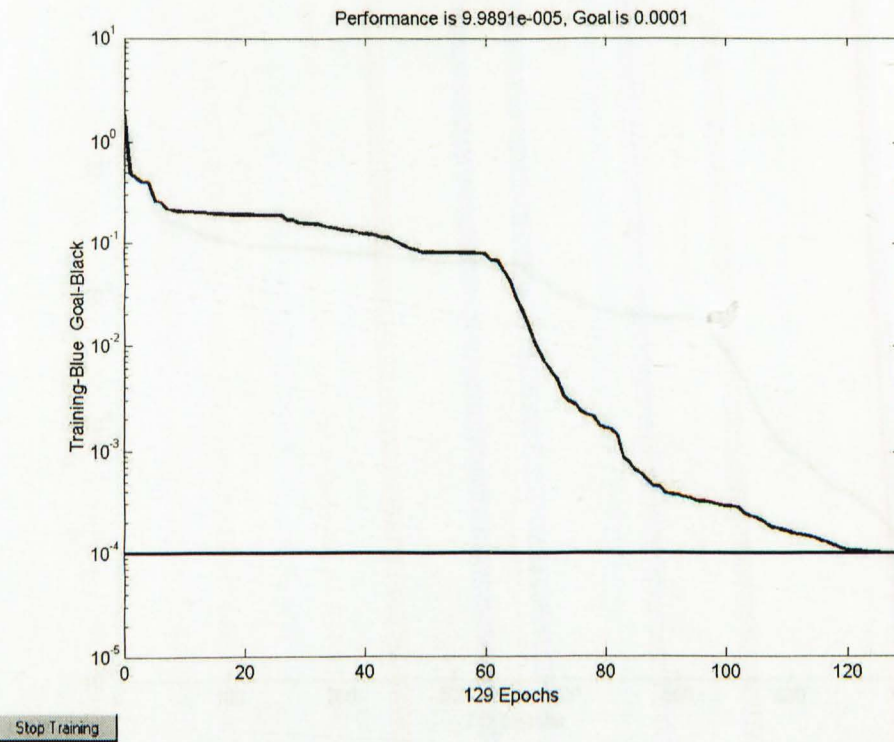


Figure B.18 The Polak-Ribiere conjugate gradient backpropagation training process of 21-3-2 neural network (Round Hole)

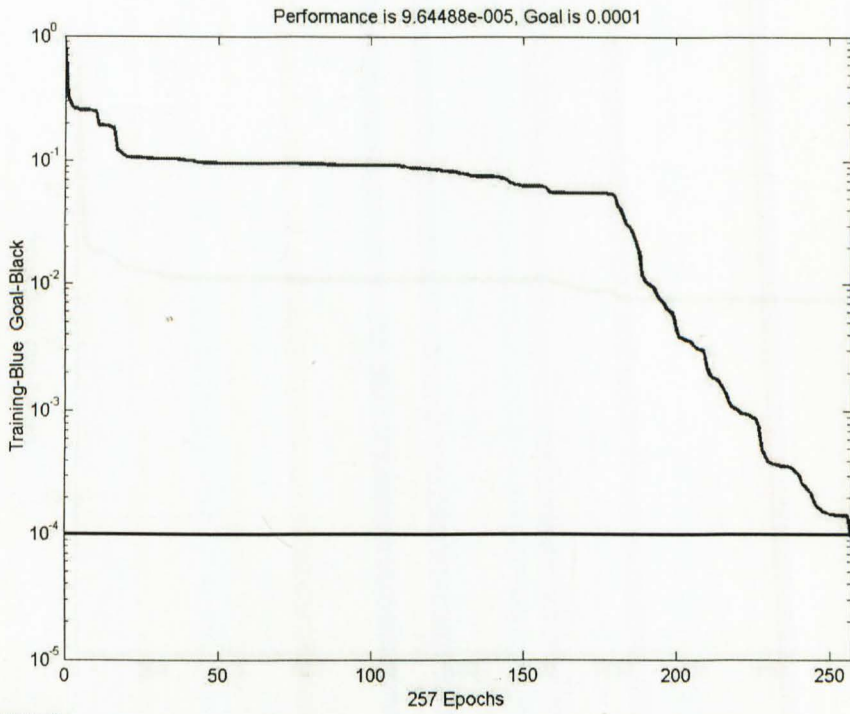


Figure B.19 The Fletcher-Powell conjugate gradient backpropagation training process of 21-2-2 neural network (Round Hole)

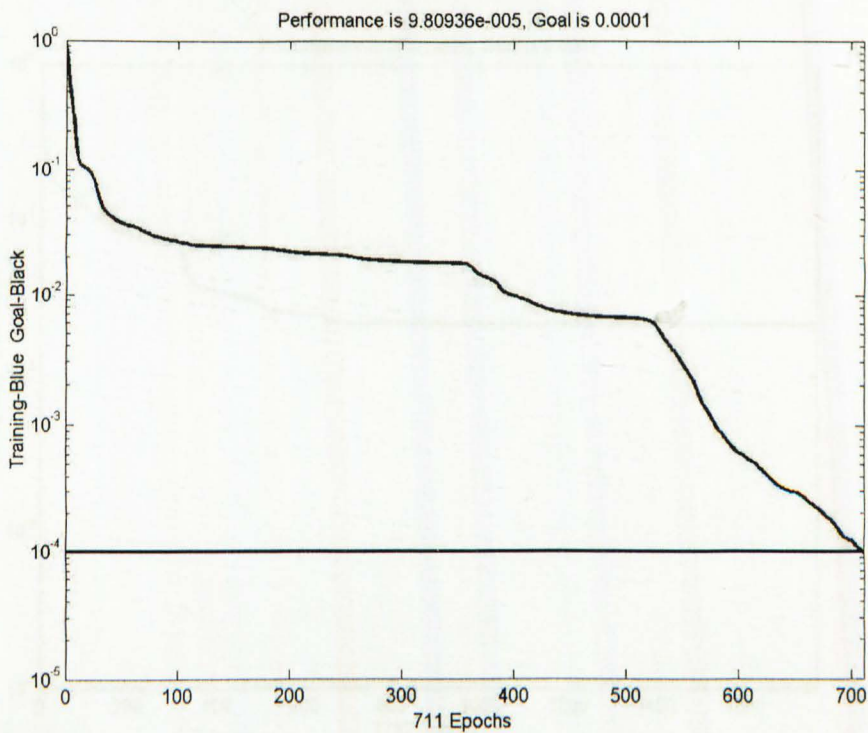


Figure B.20 The Polak-Ribiere conjugate gradient backpropagation training process of 21-6-4 neural network (Slot/Step)

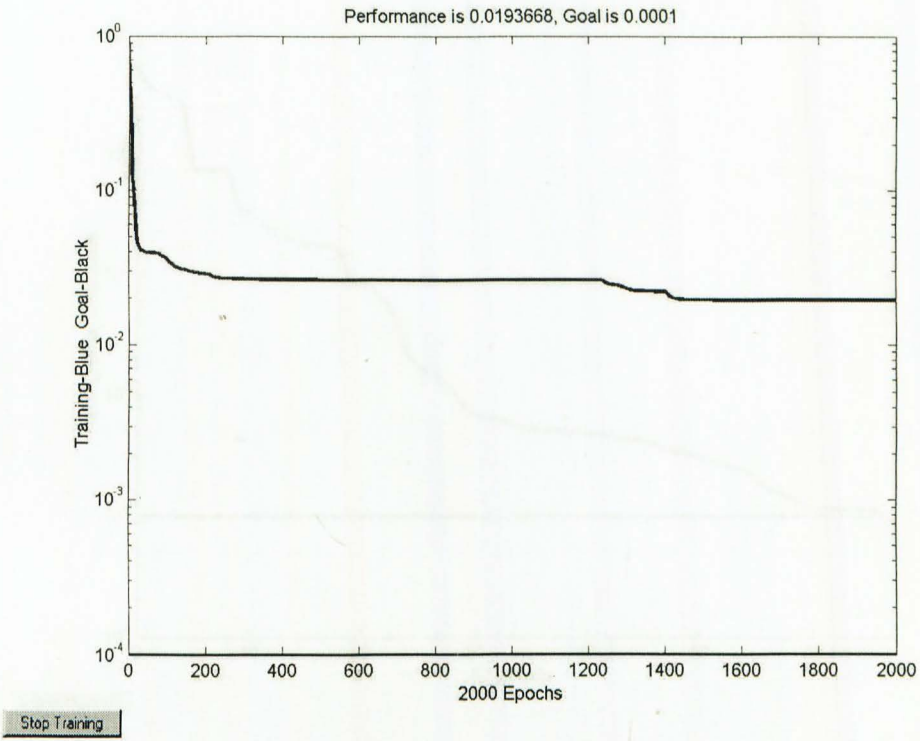


Figure B.21 The Polak-Ribiere conjugate gradient backpropagation training process of 21-5-4 neural network (Slot/Step)

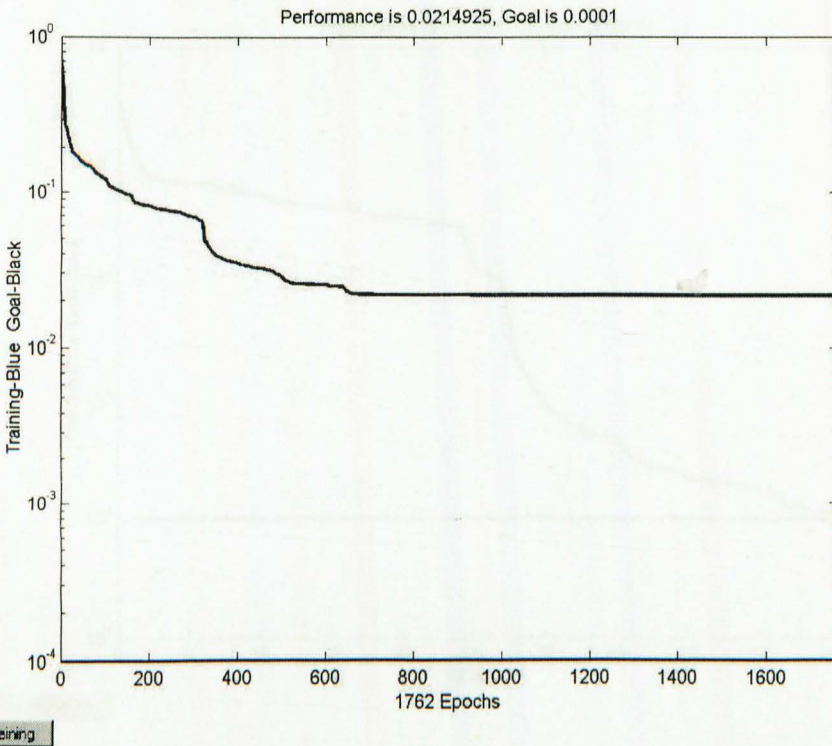


Figure B.22 The Fletcher-Powell conjugate gradient backpropagation training process of 21-6-2 neural network (Slot/Step)

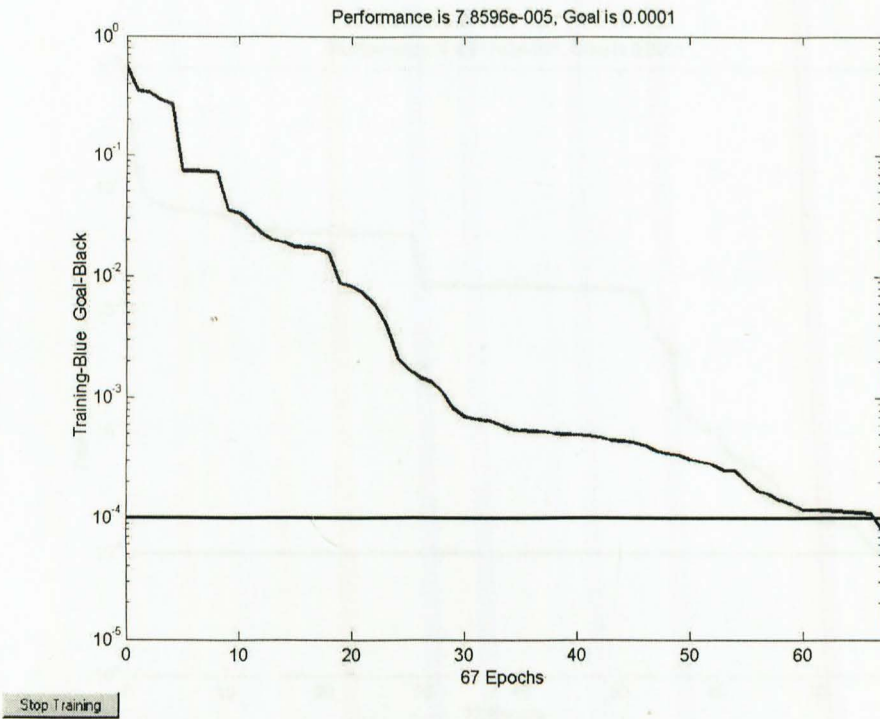


Figure B.23 The Polak-Ribiere conjugate gradient backpropagation training process of 21-2-2 neural network (Pocket)

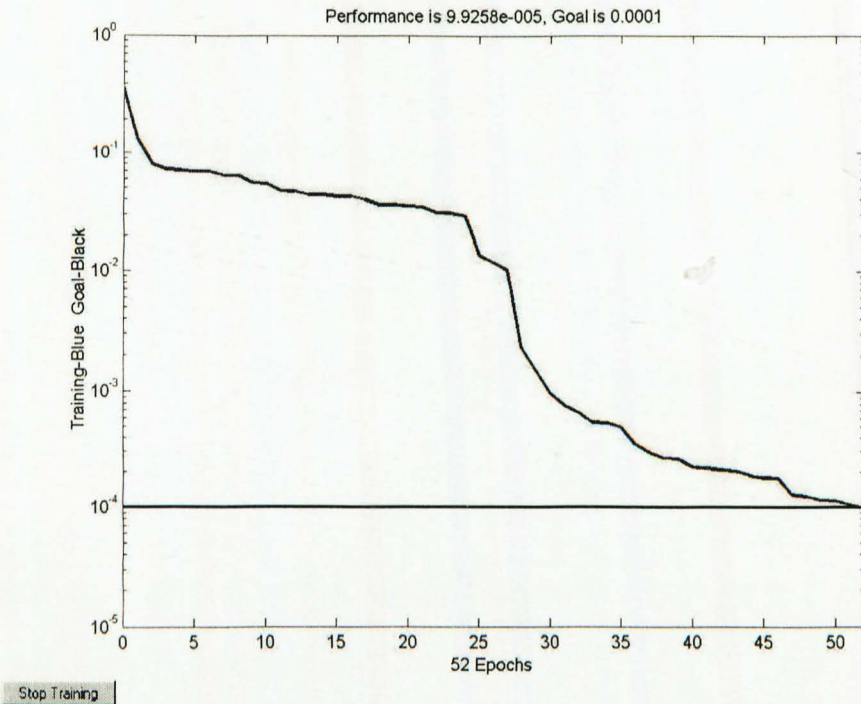


Figure B.24 The Polak-Ribiere conjugate gradient backpropagation training process of 21-1-2 neural network (Pocket)

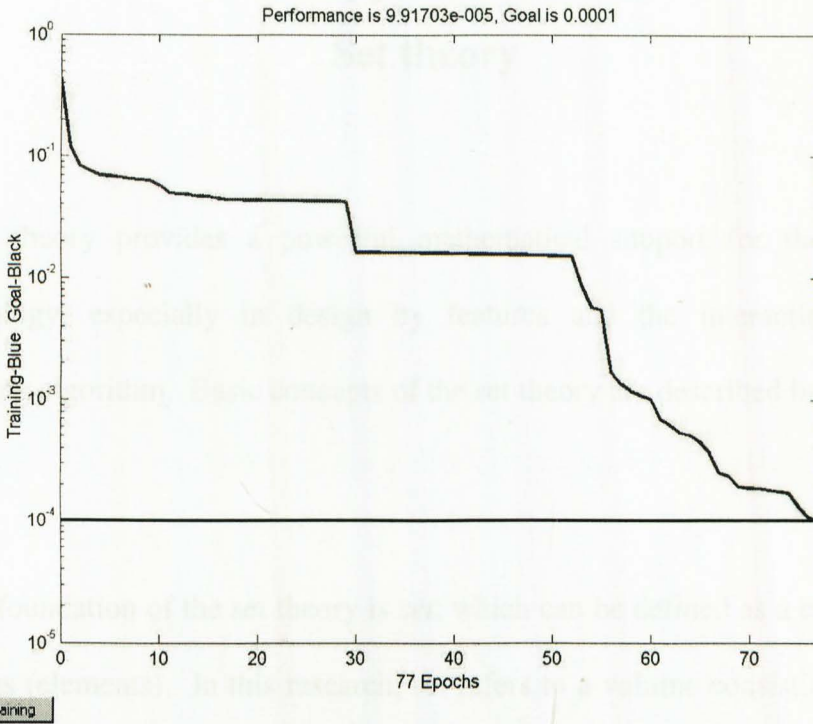


Figure B.25 The Fletcher-Powell conjugate gradient backpropagation training process of 21-1-2 neural network (Pocket)

Appendix C

Set theory

The set theory provides a powerful mathematical support for the proposed methodology, especially in design by features and the interacting features identifying algorithm. Basic concepts of the set theory are described briefly.

1) Set

The foundation of the set theory is *set*, which can be defined as a collection of things (elements). In this research, set refers to a volume consisting of faces, edges and points. To indicate that x is an element of a set A , it can be written by

$$x \in A \quad (\text{C-1})$$

Whenever x is not an element of a set A , it can be written by

$$x \notin A \quad (\text{C-2})$$

A set (say A) is called an empty set if it contains no element, that is

$$A = \phi \quad (\text{C-3})$$

If every element of set A is also an element of set B , then A is called a subset of B . It can be indicated that

$$A \subseteq B \quad (\text{C-4})$$

If set A and set B satisfy $A \subseteq B$ and $B \subseteq A$, then A and B are equal. This can be denoted by

$$A = B \quad (\text{C-5})$$

If A and B are not equal, then it can be denoted by

$$A \neq B \quad (\text{C-6})$$

If set A and set B satisfy $A \subseteq B$ and $A \neq B$, then set A is called a proper subset of set B , that is

$$A \subset B \quad (\text{C-7})$$

2) Metric space

In this research, W is a nonempty point set which represents a three-dimensional space. The points in W satisfy the following conditions

$$\text{If } x \neq y, \text{ then } d(x, y) > 0$$

$$\text{If } x = y, \text{ then } d(x, y) = 0$$

$$d(x, y) = d(y, x)$$

$$d(x, y) + d(y, z) \geq d(x, z)$$

where $d(x, y)$ is a real distance function on point x and point y .

x, y, z are the points of $W, x \in W, y \in W, z \in W$

3) Set operations

Usually, there are four basic set operations used in CAD: union \cup , intersection \cap , set complement \bar{A} and set difference $-$. Supposing two sets of A and B , then

- union \cup

The union of sets A and B is the set containing all the points that belong to either set A , set B , or both set A and set B .

$$A \cup B = \{x \mid x \in W: x \in A \text{ or } x \in B\} \quad (\text{C-8})$$

- intersection \cap

The intersection of sets A and B is the set containing all the points that belong to both set A and set B .

$$A \cap B = \{x \mid x \in W: x \in A \text{ and } x \in B\} \quad (\text{C-9})$$

- set complement \bar{A}

The set complement is the set containing all the points that do not belong to set A .

$$\bar{A} = \{x \mid x \in W: x \notin A\} \quad (\text{C-10})$$

- set difference –

The set difference of sets A and B is the set containing all the points that belongs to set A but does not belong to set B .

$$A - B = \{x \mid x \in W: x \in A \text{ and } x \notin B\} \quad (\text{C-11})$$

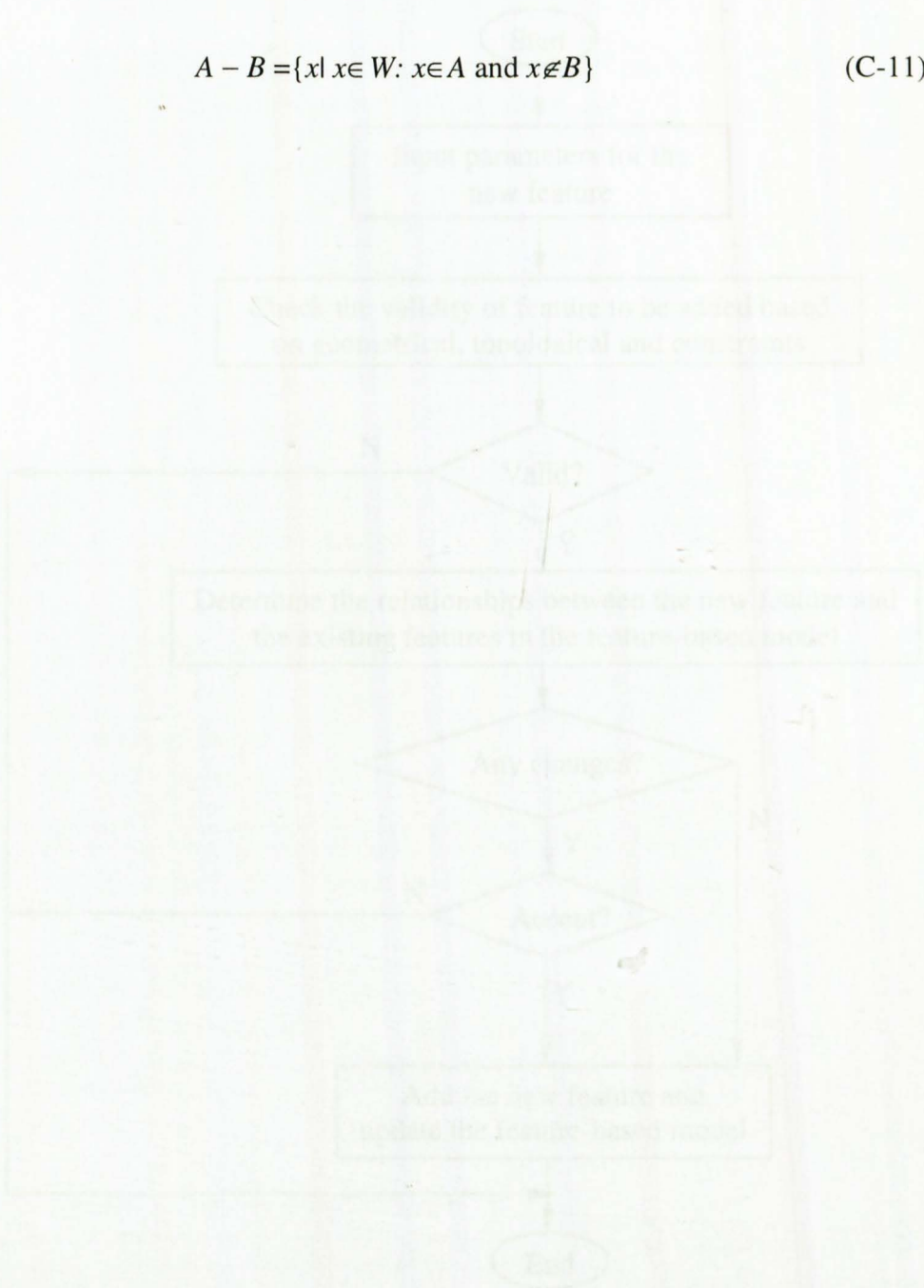


Figure D.1. Process of adding a new feature instance

Appendix D

Flowcharts

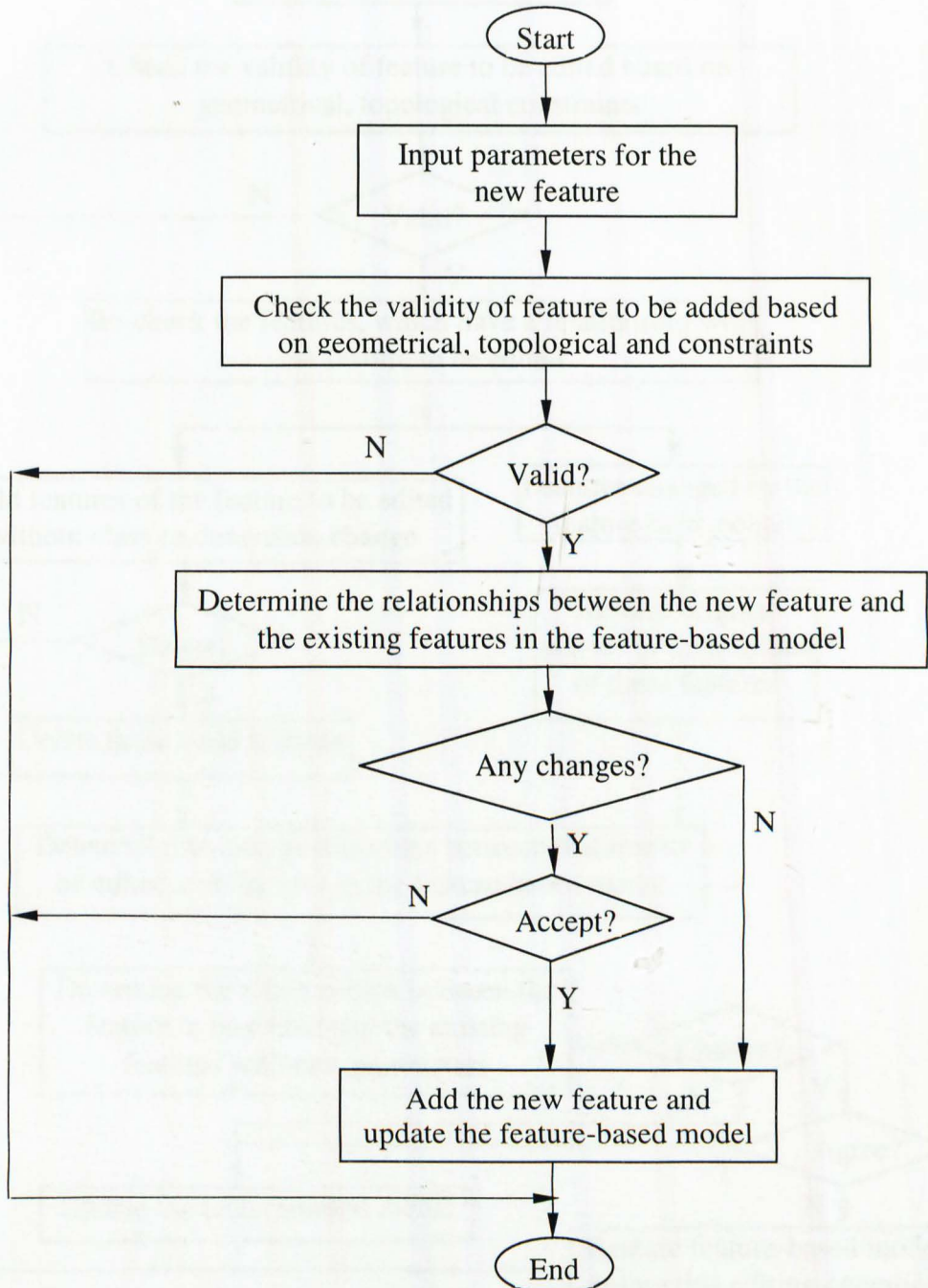


Figure D.1 Process of adding a new feature instance

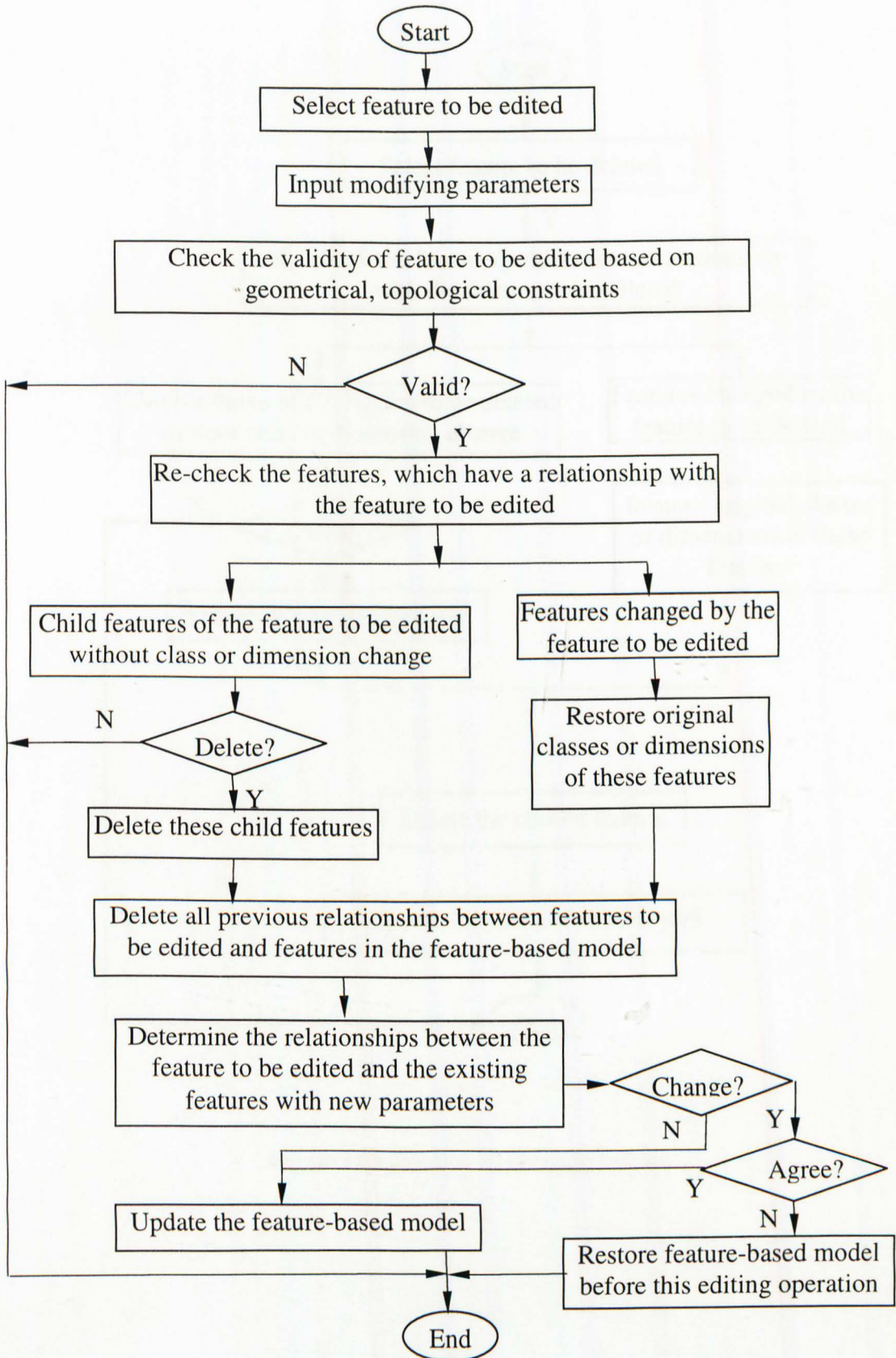


Figure D.2 Process of editing feature

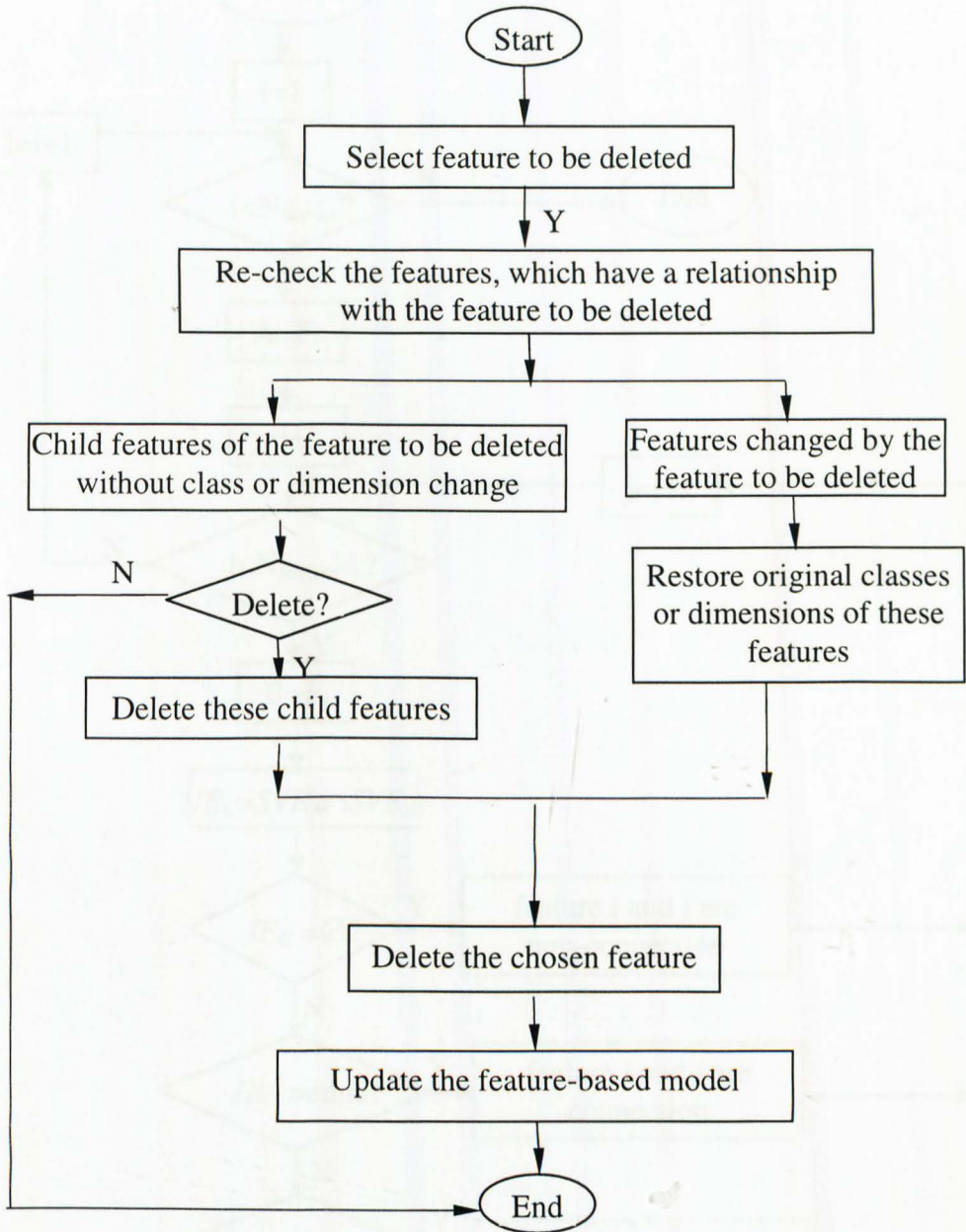


Figure D.3 Process of deleting feature

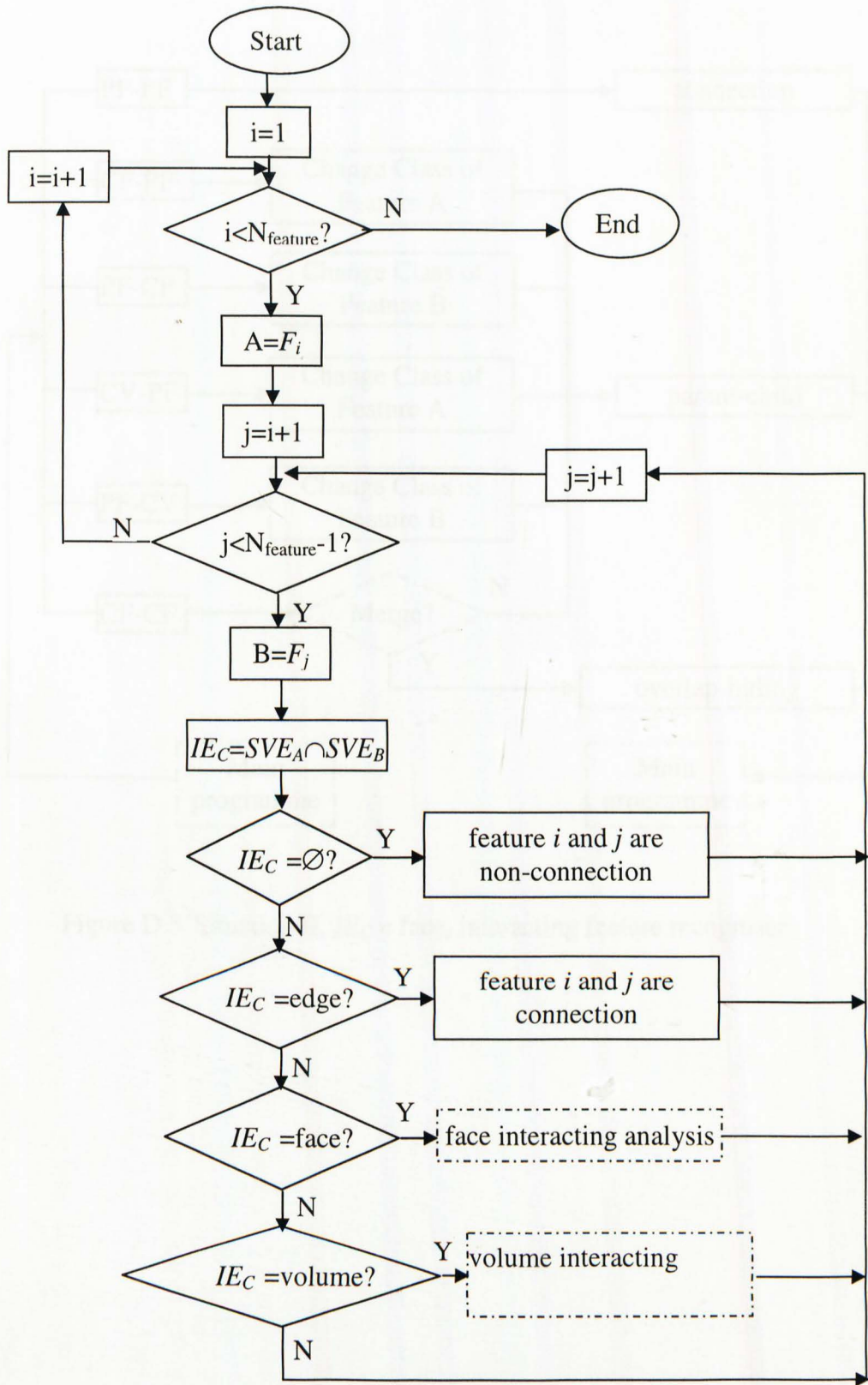


Figure D.4 Interacting feature identify algorithm

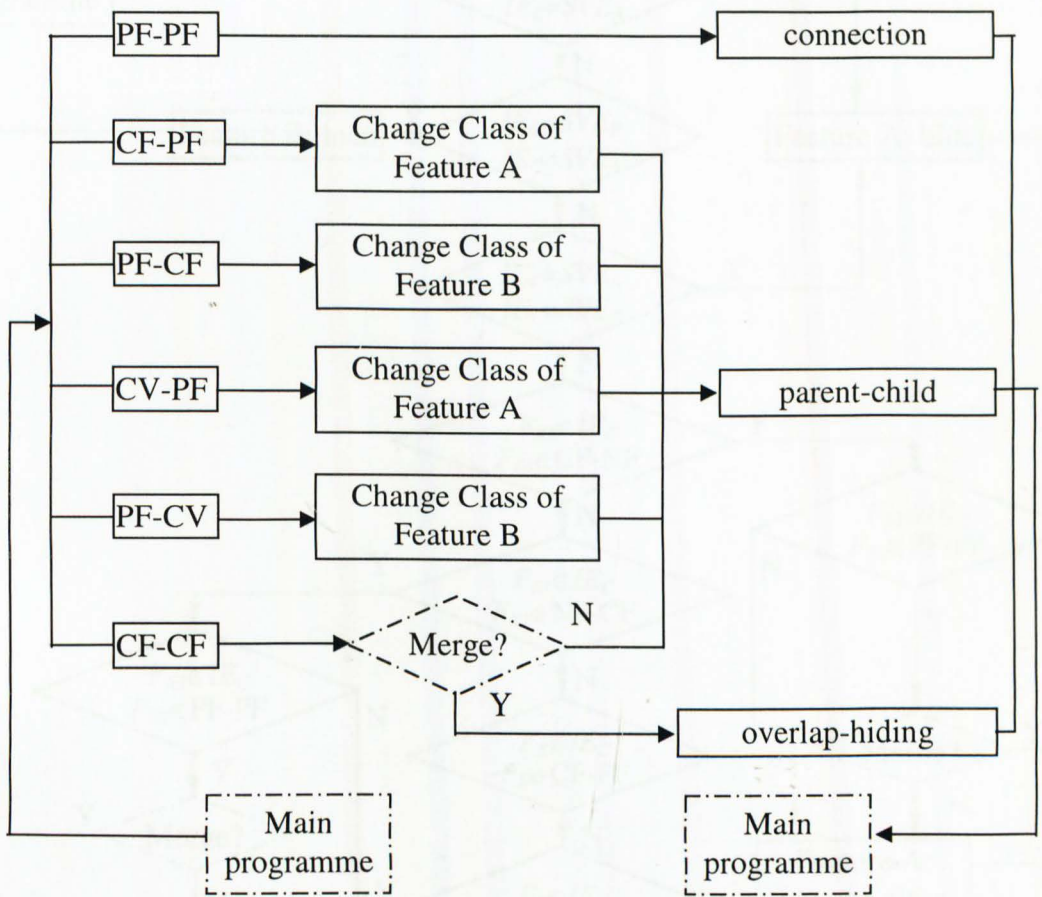


Figure D.5 Situation II, $IE_C = \text{face}$, interacting feature recogniser

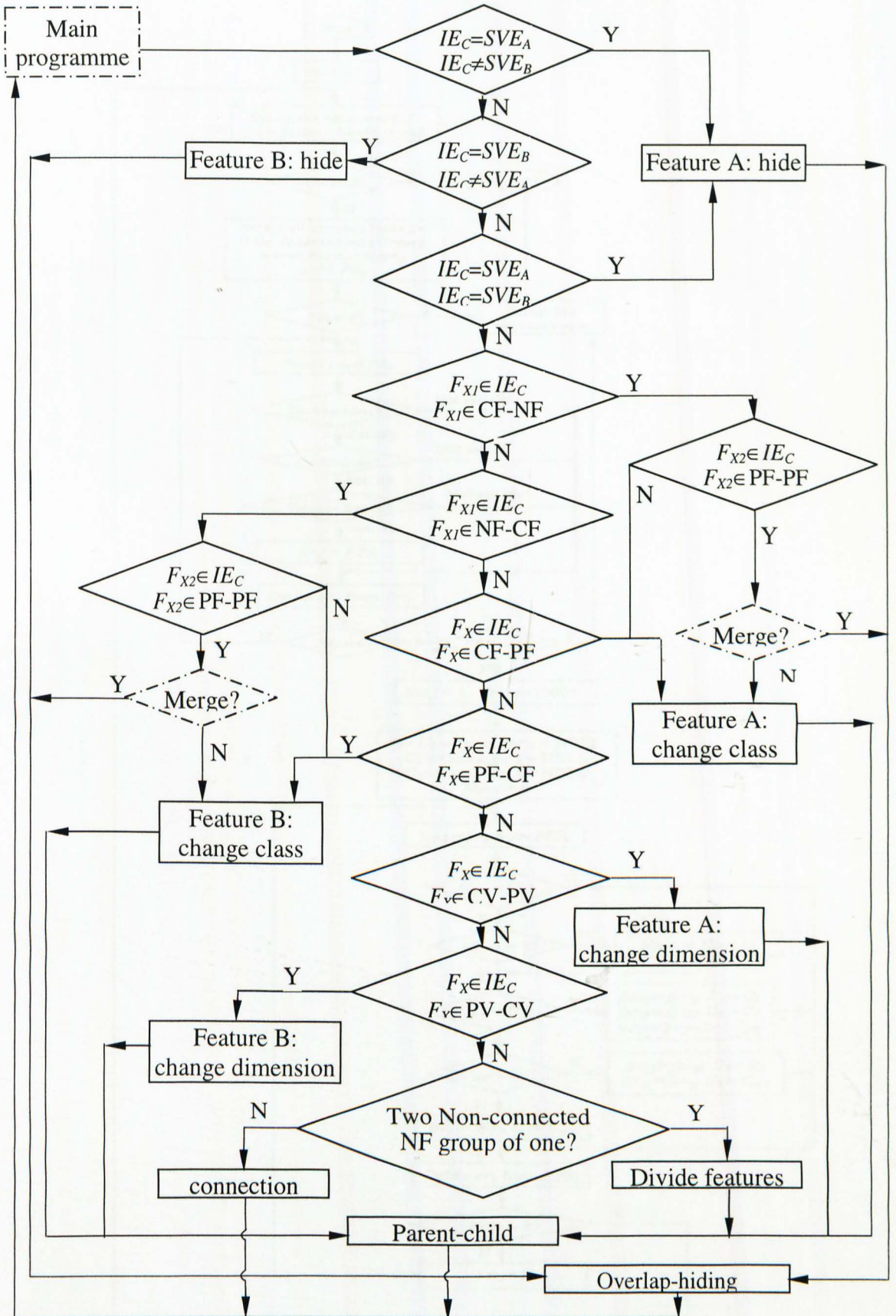


Figure D.6 Situation III, IE_C =volume, interacting feature recogniser

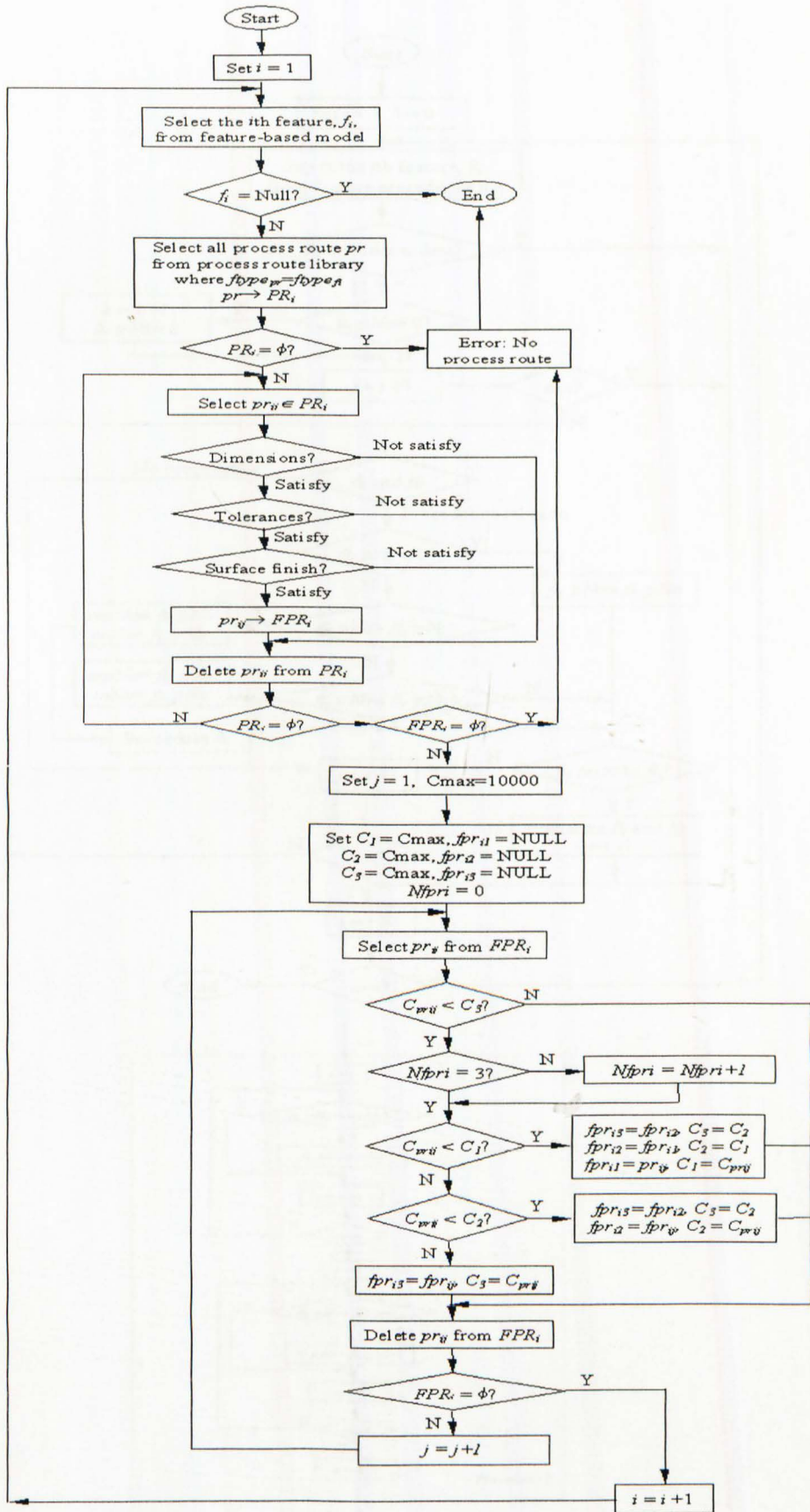


Figure D.7 Process of selecting machining operations

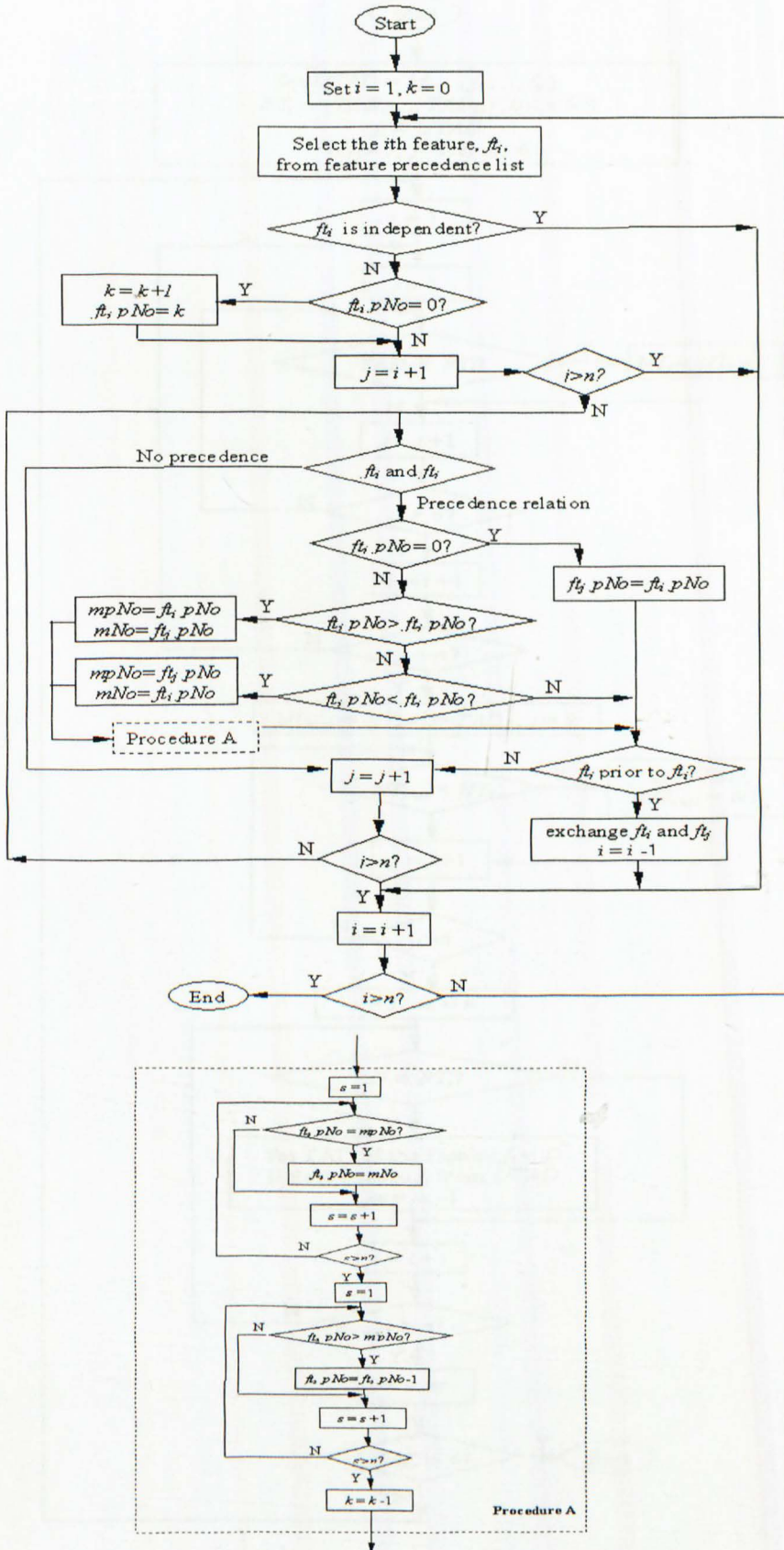


Figure D.8 Process of building a feature precedence list

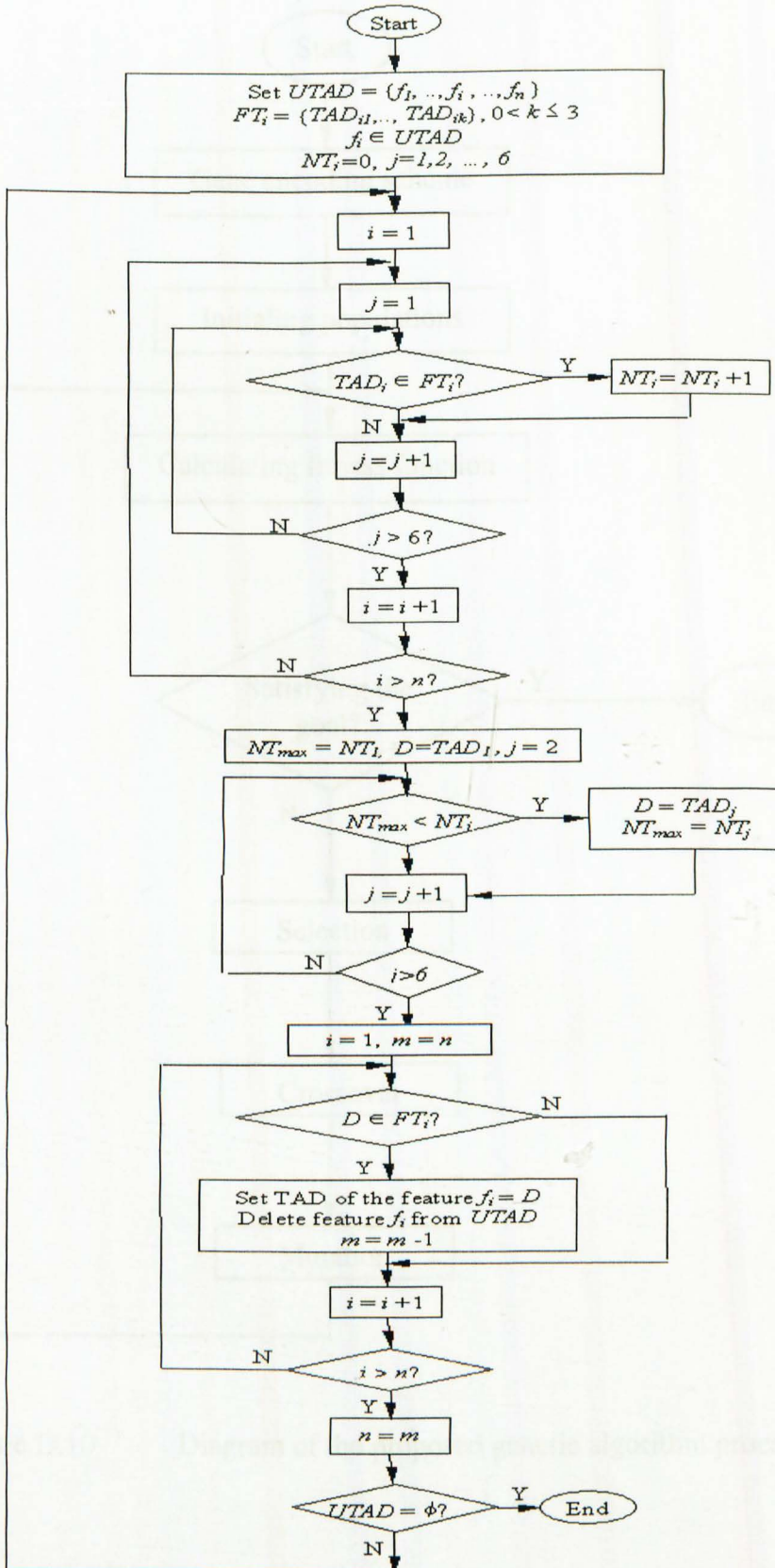


Figure D.9 Procedure for generating a tool approach direction

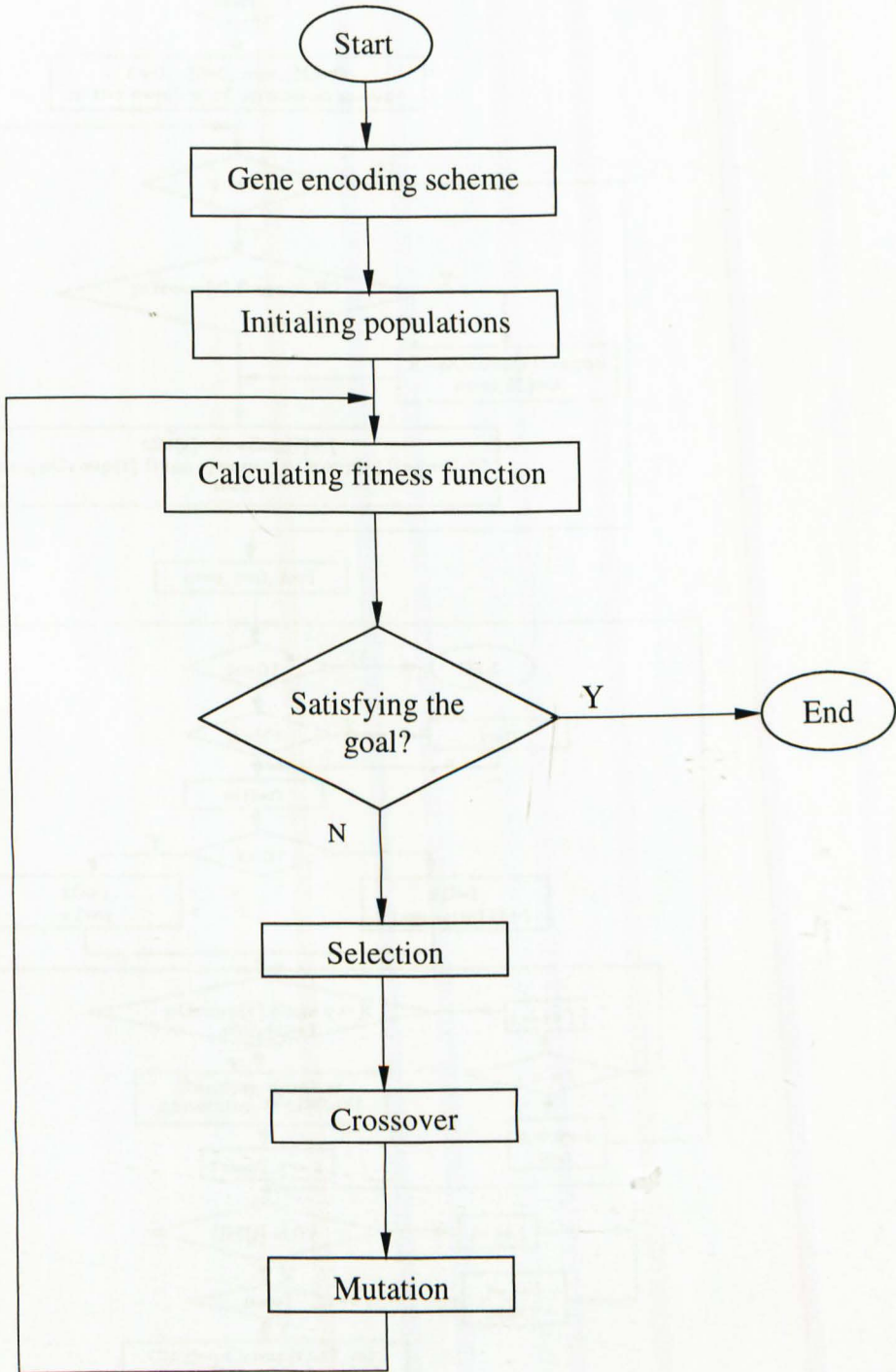
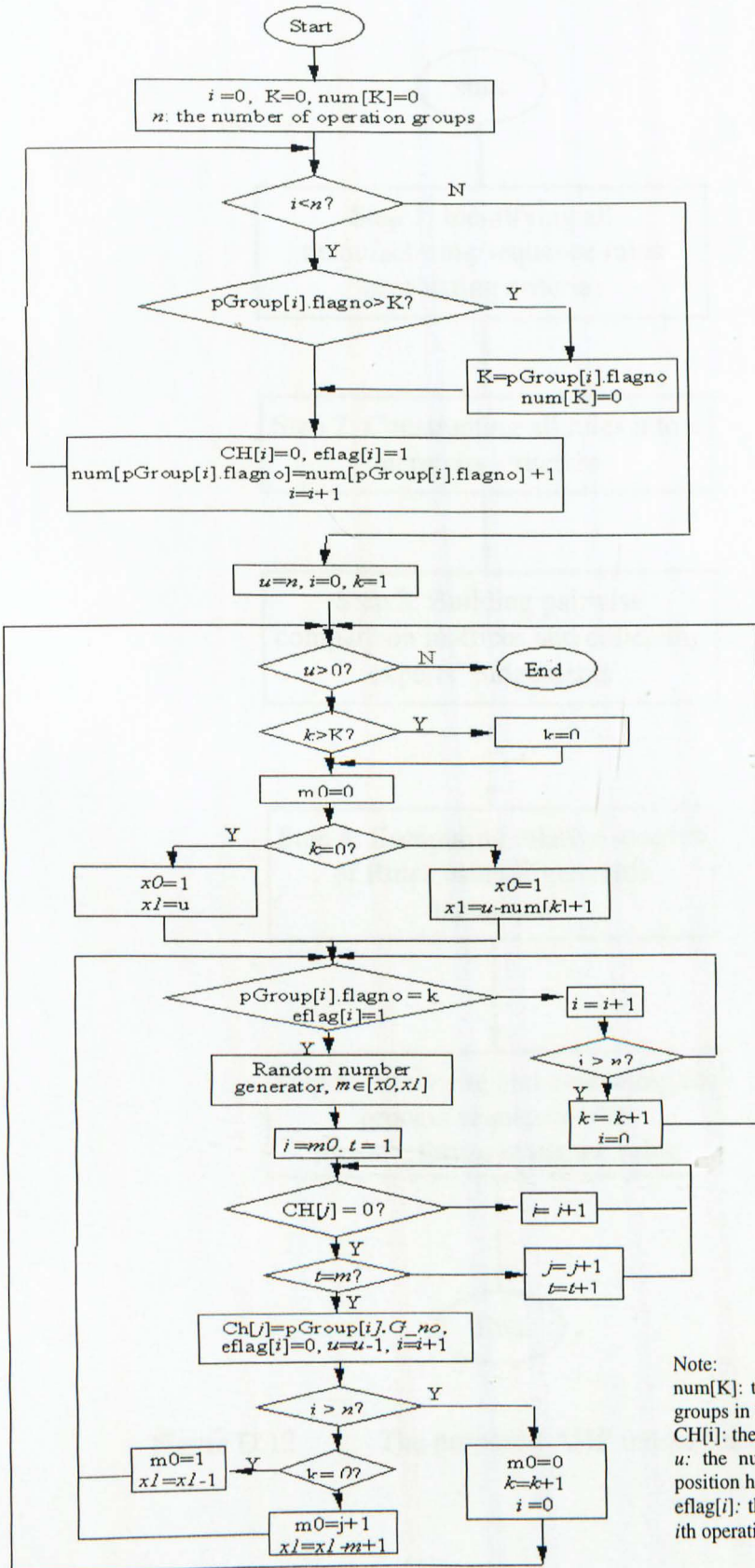


Figure D.10 Diagram of the proposed genetic algorithm process



Note:
 num[K]: the number of machining operation groups in the Kth precedence relationship
 CH[i]: the position in initial population
 u: the number of operation groups whose position has not been determined.
 eflag[i]: the flag whether the position for the ith operation groups has been determined.

Figure D.11 Initial precedence constraint algorithm

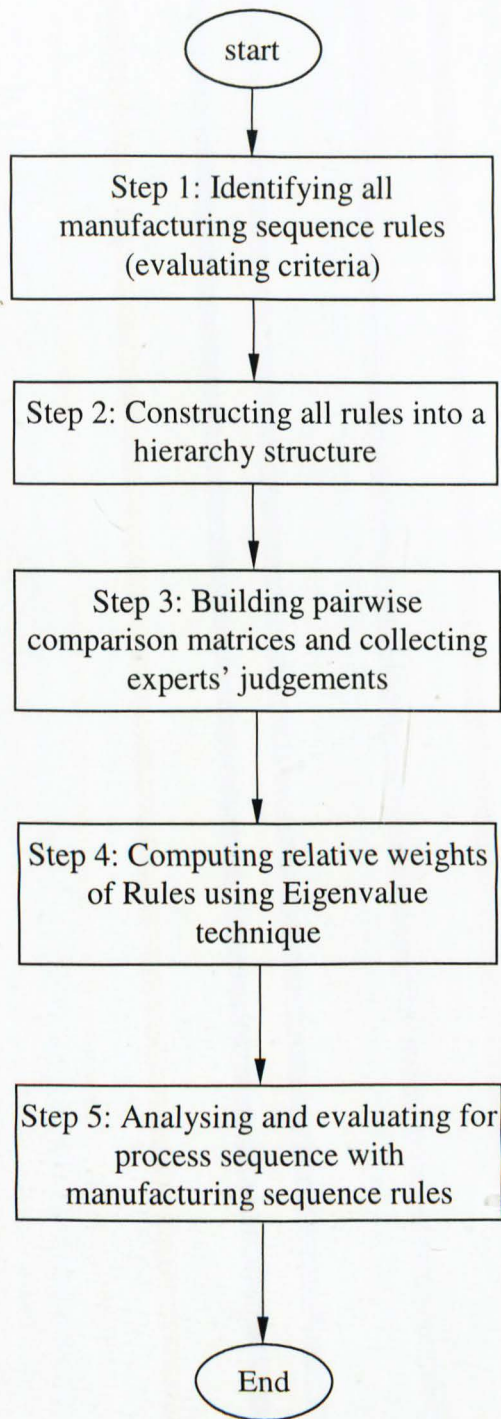


Figure D.12 The proposed AHP model for evaluation