1997

Smart Drilling: A computer-based system for planning drilling operations.

Jia. Ma

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation


This online database contains the full-text of PhD dissertations and Masters’ theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
NOTE TO USERS

The original manuscript received by UMI contains broken, slanted and or light print. All efforts were made to acquire the highest quality manuscript from the author or school. Microfilmed as received.

This reproduction is the best copy available

UMI
SMART DRILLING - A COMPUTER BASED SYSTEM

FOR

PLANNING DRILLING OPERATIONS

by

Jia Ma

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the Department of Industrial and Manufacturing System Engineering
in Partial Fulfilment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-30964-9
I hereby declare that I am the sole author of this thesis. I authorize the University of Windsor to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Jia Ma

I further authorize the University of Windsor to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Jia Ma
The University of Windsor requires the signatures of all persons using or photocopying this thesis. Please sign below, and give an address and the date.
Abstract

In this thesis, a prototype computer aided process planning system - Smart Drilling System is presented. The system will be able to automatically generate plans for multiple hole drilling process, such as tool selection, drilling sequence, and etc. The system contains four modules: (1) Tool selection module, (2) Machining condition design module, (3) NC code generation module, and (4) Simulation module. In the tool selection module, a generative searching and planning algorithm is applied. Fuzzy logic was utilized to make decisions in tool selection. The module can group the drills according to their size and shape, and select optimal drills that will drill maximum number of holes. This will guarantee the minimum tool changes and reduce the machining time. It also can minimize the drill travelling time by ensuring the drills always finish the nearest holes first. The machining condition design module is implemented to determine machining conditions (speed, feedrate, etc.), based on the minimization of production cost. The NC code generation module can automatically generate the NC code, which is APT format. Finally, the simulation module is able to display the drilling operation by using VERICUT®.

In total, 20 simulation examples were tested by using the Smart Drilling System. From the simulation results, the system could save approximately 50% cycle time compared to sequence drilling planning.
DEDICATION

To my family

For their true love
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to Dr. R. Du for his guidance and support throughout the study. Under his supervision, I have been able to focus my efforts on the CAPP system development.

Deep appreciation are extended to Dr. S.P. Dutta who has been encouraging and inspiring me throughout my graduate study. I would like to thank my committee members, Dr. W.P. North and Dr. S.Taboun for their advice. Much thanks to Ms. J. Mummy, secretary of Industrial and Manufacturing Systems Engineering, for her assistance and personal encouragement.

I would like to thank my parents, my wife, and my sister. I own the sincere debt of gratitude for their support and encouragement.

Finally, I would like to thank all my friends who have been constantly supporting me through their concern in my life.
TABLE OF CONTENTS

ABSTRACT vi
DEDICATION vii
ACKNOWLEDGEMENTS viii
LIST OF FIGURES xi
LIST OF TABLES xiii
NOMENCLATURE xiv

CHAPTER

I. INTRODUCTION 1
1.1 Background and Overview 1
1.2 Organization of the Thesis 4

II. LITERATURE REVIEW 5
2.1 Reviews on CAPP 5
2.1.1 Background 5
2.1.2 Role of process planning 6
2.1.3 Approaches of CAPP 8
2.1.4 Implementation techniques 12
2.2 Reviews on Fuzzy System 18
2.2.1 Overview 18
2.2.2 Procedure for developing fuzzy system 21

II.3 Motivation of the Research 22

III. METHODS USED IN THE SYSTEM 23
3.1 Decision Logic 23
3.1.1 Decision trees 24
3.1.2 Decision tables 25
3.2 Fuzzy Logic 27
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Various types of drills and drilling operations</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Holes with different diameters and shapes</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>A schematic of 'bottom-up' CAPP system</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>An architecture of a 'top-down' system</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Architecture of a general KBES</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>A decision tree in proposed system</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>An example of Tmatrix</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Linguistic and numerical values of small</td>
<td>30</td>
</tr>
<tr>
<td>3.4</td>
<td>Representation of a function and its fuzzy graph</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>Types of approximation: functional, relational, and fuzzy graph</td>
<td>35</td>
</tr>
<tr>
<td>3.6</td>
<td>Intersection of fuzzy graphs f^ and g^</td>
<td>37</td>
</tr>
<tr>
<td>3.7</td>
<td>Intersection of f^ and ( \overline{A} )</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>The meaning of the degree of match between A and A_i</td>
<td>39</td>
</tr>
<tr>
<td>3.9</td>
<td>Intersection of fuzzy sets F and G whose membership functions are</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>represented as fuzzy graph</td>
<td></td>
</tr>
<tr>
<td>3.10</td>
<td>A relation with fuzzy-valued attributes</td>
<td>42</td>
</tr>
<tr>
<td>3.11</td>
<td>Diagnostic tableau</td>
<td>42</td>
</tr>
<tr>
<td>3.12</td>
<td>An example of using fuzzy decision table</td>
<td>50</td>
</tr>
</tbody>
</table>
3.13 Calculation of alternative drill diameter
3.14 An example of Tmatrix contains multiple predrills
3.15 An example of determination of multiple predrills
3.16 Determination of drilling sequence
4.1 Different type of holes
4.2 Description of the procedure of smart drilling
4.3 Details of sorting & listing module
4.4 Details of searching & matching module
4.5 Structure of smart drilling system
4.6 An example of generated APT code
4.7 Statistic results of drilling simulations in terms of cutting time
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Generic categories of fuzzy system applications</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>A partial list of application areas of fuzzy systems</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>A fuzzy relation $f'$</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Status results of smart drilling simulation</td>
<td>78</td>
</tr>
</tbody>
</table>
NOMENCLATURE

A, B = fuzzy sets
U, V = universe sets
u, v = elements in universe
m(u,v) = membership function
X, Y = fuzzy subsets
f * = fuzzy graph
(u ∈ U) = u belongs to the set of U
∧ = minimum operator
∨ = maximum operator
× = Cartesian product
* = operation of nondecreasing
∩ = operation of intersection
∪ = operation of union
D = diameter of drills (in.)
f = feed rate (mm/rev)
N = rpm of the drill (rev/min.)
TM = total manufacturing time (sec.)
t_MD = total machining time for drilling (sec.)
t_MV = total time for tool movements (sec.)
t_TC = total time for tool exchange (sec.)
l = drilling depth (in.)

k = index for number of drilling cycles

i = index for tool

tdist( ) = function to determine shortest path

\( \phi(j) \) = end position of hole j drilled by tool i

\( \phi(j + 1) \) = start position of hole j+1 will be drilled by tool i

\( n_{TC} \) = number of tool changes

\( t_{TC} \) = total tool change time (sec.)

\( t_{TCi} \) = tool change time (sec.)

\( C_M \) = manufacturing cost ($)

\( C_T' \) = tool cost of ith tool ($/tool)

\( T_i \) = tool life span of ith tool (hours)

\( C_{tot} \) = the total cost ($)

\( \gamma \) : weighting factor [price unit per time unit]

TM = total manufacturing time (sec.)
CHAPTER I
INTRODUCTION

1.1 Background and Overview

Of all the metal cutting operations, drilling is the most commonly used one, and accounts for one-third of all the metal cutting operations in industry (Ogawa 1985). It is the quickest and the most economical method for hole production. In general, drilling is applied in situations where precision work is not of prime concern and its main advantages include high material removal rate and longer tool life. Also, it is a preliminary operation to reaming, boring or grinding, where final finishing and sizing takes place.

Drills usually have a high length-to-diameter ratio, hence are capable of producing deep holes. However, they are somewhat flexible, depending on their diameter, and should be used with care in order to drill holes accurately and to prevent the drill from breaking. Various types of drills and drilling operations are shown in Figure 1.1(Kalpakjian, 1992).

![Various types of drills and drilling operations](image)

Figure 1.1 Various types of drills and drilling operations
This research focuses on the multiple hole drilling problem, which is common in practical drilling on the shop floor. There could be many holes in the part, shown in Figure 1.2. Actually, there are many factors to be considered for the drilling processes; one of them is process planning. Obviously, it's very important to make sure that the drilling process is efficient. Usually, this work is normally done by people with experience. However, with the rapid advance of computer techniques, Automated Process Planning, has become technically possible and economically profitable. Nowadays Computer Aided Process Planning (CAPP) plays an important role in modern industry.

Figure 1.2 Holes with different diameters and shapes

The use of computers for design and manufacturing is having a dramatic impact on the development of production automation technologies. Computers are helping to reduce the amount of lead time required, and possibly cost, from part conceptualization to actual
part production. This process is made possible because certain individual functions relating to the design and manufacturing activities are being computerized.

To integrate and process information from the design function to manufacturing activities may generally involve three main aspects: CAD, CAPP and CAM which are the acronyms for computer-aided design, computer-aided process planning and computer-aided manufacturing. CAD is concerned with the design, modeling and analysis of a given part. When the design is complete, the CAPP system will then address and plan the allocation of manufacturing tasks for a given part. The main CAM role is to support the production operations directly or indirectly in the making of the part.

CAPP has long been considered as an essential link for the integration of CAD and CAM activities. Recently, the emergence of concurrent engineering has placed an even greater emphasis on the automation of process planning, since the product design in a concurrent engineering environment imposes the timely availability of manufacturing information such as manufacturability, machining time, cost, etc.

Over the last twenty years, a large number of CAPP systems have been developed using different approaches, such as variant approach and generative approach. Various successes in automating portions of process planning tasks have been achieved. At the beginning of the 1980s Artificial Intelligence (AI) techniques were introduced in the CAPP area. Many CAPP systems were implemented by AI techniques, which were usually entitled either as 'knowledge-based' systems or 'expert' systems. Each of them has its own advantages and disadvantages. A lack of skilled process planners has occurred in some industrialized countries, such as in the USA, the UK etc., which gives impetus to the
development. More and more industrial companies have acquired CAPP systems for integration of design and production and to compensate for the shortage of skilled process planners. In spite of the fact that many CAPP systems have been developed, their effectiveness is still far from satisfactory, and many large companies have had to establish their own research groups to develop their own CAPP systems. Small and medium sized companies can afford only existing CAPP systems which have been developed by research organizations or universities.

In this research, CAPP for multiple hole drilling is presented, which focuses on the following items of process planning:

- selection of drills to reduce tool change
- arrangement of drilling sequence to reduce tool traveling
- consideration of the optimal drilling condition to minimize cutting cost.

1.2 Organization of the Thesis

This thesis consists of five chapters. Chapter II presents a literature review of the related research publications involving CAPP and fuzzy systems. Chapter III describes the method used in the system for selecting drills, planning drill traveling and designing cutting conditions. Chapter IV describes the details of the development of the smart drilling system and gives some simulation examples. Finally, conclusion and future work are discussed in Chapter V.
CHAPTER II

LITERATURE REVIEW

2.1 Reviews on CAPP

2.1.1 Background

In 1965 Niebel first presented the idea of using the speed and consistency of the computer to assist in the determination of process planning (Niebel 1965). After that Schenk (1966) discussed the feasibility of automated process planning in his Ph.D dissertation at Purdue University. Despite the early recognition of the possibility of extracting operations and processing sequences from part geometry as described in CAD, computer aided process planning has not been broadly addressed until the beginning of 1970s. This is probably due to the fact that the computer capabilities of both hardware and software were limited and manufacturing engineering was more or less isolated from computer aided techniques at that time. 1976 probably was the first ‘harvest’ year in the CAPP area. The CAPP (this is the actual system name) system, the first variant system, was developed and presented at the ‘1976 NC Conference’ (Link 1976). In the same year another system MIPLAN developed by the OIR (Organization of Industrial Research) was presented (Houtzeel 1976). Subsequently Wysk (1977) presented a generative system for detailed process selection titled APPAS in his Ph.D dissertation. Since then, CAPP has begun to be widely addressed. In the meantime, the number of skilled process planners has declined in many industrial countries. In order to replenish the shortage of skilled process planners and increase their competitiveness, companies try to adopt flexible automation.
CAPP as a main element in the integration of design and production has not kept pace with the development of CAD and CAM. This situation has made process planning a bottle neck in the manufacturing process (Allen 1987). Thus more and more effort has been applied in the CAPP area and numerous CAPP systems have been reported.

2.1.2 Role of process planning

Process planning has been defined by the Society of Manufacturing Engineers as "the systematic determination of the methods by which a product is to be manufactured economically and competitively" (Davies 1984). Usually the task of process planning involves a series of steps. The first consideration is the interpretation of the design data which is usually displayed by blueprints in the traditional way, and now by computer design data displayed on a CAD system. In this stage the requirements of products such as batch size, geometric configuration, raw material property, dimension tolerances, surface roughness, heat treatment and hardness, as well as some special requirements is studied and interpreted. Then, according to the interpretation, the machining process which is usually based on a company specific strategy is examined. This strategy consists of turning, milling, drilling, grinding, etc. Next, the machining centers which usually can carry out one or multiple machining processes have to be selected. The selection should consider availability, process capability (size, accuracy, etc.), range of machining operations, production rate, etc. Then the operation sequence must be determined which is usually based on a company-specific strategy. This strategy consists of comprehensive operations for a defined group of parts. Each operation is described by a selection criteria which depends on the shape and dimension of the part. Then the clamping devices and
datum reference surfaces have to be selected to hold the components while they are being machined and to allow access for machining. Often the situation is that no proper holding devices exist, and the necessary fixtures have to be designed by the process planner himself. The appropriate tools and cutting conditions such as depth of cut, feed and speed rate have to be determined. The overall machining and non-machining times have to be calculated, including batch set up time, load and unload, tool changing and inspection times. Sometimes the cost of the process is included. Finally, process sheets, operation sheets, (sometimes referred to as operation sheets), planning sheets or route sheets, and part programs are made and edited. The checking for syntax and path errors should be considered in this stage. Significant economical benefits could be obtained if optimum production concepts are adopted throughout the CAPP process (Lye 1992). Process planning can be itemized as follows (Chang 1991):

1. Interpretation of product design data.
2. Selection of machining processes.
3. Selection of machine tools.
4. Determination of fixtures and datum surfaces.
5. Sequencing the operations.
6. Selection of inspection devices.
7. Determination of production tolerances.
8. Determination of the proper cutting conditions.
9. Calculation of the overall times.
10. Generating of process sheets including NC data.
Part of the above items of process planning are discussed in smart drilling system.
such as selection of machining processes, sequencing the operations, determination of
proper cutting conditions, calculation of the time and generating NC data.

2.1.3 Approaches of CAPP

Two approaches to computer aided process planning are traditionally recognized,
the variant approach and the generative approach. However, with the rapid development
of new techniques, many CAPP systems do not exactly fit either classification but
combine both approaches.

2.1.3.1 The variant approach

The variant approach to process planning is comparable with the traditional
manual approach where a process plan for a new part is created by recalling,
identifying, and retrieving an existing plan for a similar part, and making necessary
modifications for the new part. In some variant systems parts are grouped into a
number of part families, characterized by similarities in the standard process plan,
which, including all possible operations for the family, is stored in the system.
Through classification and coding, a code is built up by answering a number of
predefined questions. These codes are often used to identify the part family and the
associated standard plan. The standard plan is retrieved and edited for the new
part. The variant approach is widely used, e.g. CAPP( a real computer aided
process planning system) (Link 1976), MIPLAN (Houtzeel 1980), etc. In
comparison with manually performed process planning, the variant approach is
highly advantageous in that activities and decisions require less time and labor.
Also, procedures can be standardized by incorporating a planner’s manufacturing knowledge and structuring it to a company’s specific needs. Therefore, variant systems can organize and store completed plans and manufacturing knowledge from which process plans can be quickly evaluated. However, there are difficulties in maintaining consistency in editing practices, and inability to adequately accommodate various combinations of geometry, size, precision, material, quality, and shop loading. The biggest disadvantage is that the quality of the process plan still depends on the knowledge of a process planner. The computer is just a tool to assist in manual process planning activities. However, the variant approach is still popular.

The main reasons probably are:

1. The investment is less and the development time is shorter, especially for medium sized companies which want to establish their own research groups.

2. The development costs and hardware costs are lower, especially for some small companies where the products do not vary very much and who will still have process planners.

2.1.3.2 The generative approach

In the generative approach, process plans are generated by means of decision logics, formulae, technology algorithms, and geometry-based data to perform uniquely the many processing decisions for converting a part from raw material to a final state. The rules of manufacturing and the equipment capabilities are stored in a computer system. When using the system, a specific process plan for a specific part can be generated.
without any involvement of a process planner. For generative systems, input can come either as a text input where the user answers a number of questions in an English dialogue (defined as interactive input), or as graphic input where the data is gathered from a CAD module (defined as interface input). So far the former is more common in existing CAPP systems, while the latter is still a fairly undeveloped area because of its complexity. Nevertheless, interface input is necessary to enable an integrated manufacturing system. Much effort has been expanded to interface CAPP with CAD. The terms, feature recognition, feature extraction, feature refinement, and geometry reasoning have been used to denote the study which has been discussed in detail in Chang (1991).

The generative approach is complex and a generative CAPP system is difficult to develop. At the beginning, it was argued that this type of system was too complex to ever be computerized. However, with the rapid development of AI techniques, the success of applying AI techniques in other areas has greatly encouraged the utilization of the AI techniques in process planning. This effort has given initial results which indicate that generative systems are desirable and promising.

Several generative process planning systems have already been developed such as APPAS (Wysk 1977), CMPP (Waldman 1983), EXCAP (Davies and Darbyshire 1984), XPLAN (Lenau and Alting 1986), and so on. The biggest advantage of the generative approach is that the process is consistent and fully automated. This kind of system is mostly oriented toward large companies and research organizations since they can afford the investment on a long term project. For companies which have a number of products in small lot sizes the generative approach is especially important. Five alternative approaches
to generative process planning are discussed in detail in Allen (1987): decision tables; decision trees/decision tables; axiomatic; rule-based decision tree; and constraint-based approaches.

2.1.3.3 The semi-generative approach

The semi-generative approach is an interim approach and much effort has gone into “semi-generative” CAPP systems. These serve to reduce user interaction through such features as standard operation sequences, decision tables and mathematical formulas. These schemes are not completely generative, but they can be extremely useful in terms of time and cost savings in the manufacturing environment. So far, under the assumption that no consideration generative process planning systems exist, the semi-generative approach, as a stand-alone process planning approach, is understandable. The term ‘semi-generative’ may be defined as a combination of the generative and the variant approach, where a pre-process plan is developed and modified before the plan is utilized in a real production environment. It means that the decision logic, formulae, and technological algorithms as well as the geometry-based coding scheme for translating physical features (such as features size, locations, etc.) are built into the systems. At first, a final process plan has to be examined and errors corrected if it does not fit into the real production environment. It may be a good idea to break a generative system down in a generating planning stage and modifying stage to correct the plans which may be in the variant approach. From a research point of view the semi-generative system may not be the desired direction, but it increases the system’s competitiveness on the market. Industrial applications of such systems have the following advantages:
(a) speed up automatic production
(b) reduce the process planners’ participation
(c) ensure the quality of process plan

2.1.4 Implementation techniques

Implementation of CAPP systems covers broadly current technologies. There are the following popular techniques:

**Group technology**

The concept of Group Technology (GT) was first discussed about 50 years ago by Sokolowski at Leningrad in the soviet union (Tuffentsammer and Arndt 1983). Unfortunately, this idea was not widely adopted until the beginning of the 1970s. Since computer aided technologies and Flexible Manufacturing Systems (FMS) were introduced into the manufacturing industry, GT has become recognized and are widely utilized in CAPP systems. Usually GT can be defined as the philosophy of studying a large population of apparently different items and then dividing them into groups of items having the same or similar characteristics. Since the mid-70s, Allen of Brigham Young University and Ham of Penn. State University have pushed the idea of GT for American industry. Now GT is considered quite mature in the CAPP area. The typical utilization of GT is in the part family concept where coding and classifying of the part are used(Chang 1991).

In general, five methods are used to form part families:

1. Manual/visual search: oracular grouping, better known as ‘eye balling’ is a popular way of grouping parts, but it has many limitations.
2. Nomenclatures/function: names are a useful way of identifying people, but have several limitations when it comes to identify parts.

3. Production flow analysis: group all the parts that are to be produced by a given group of machine tools, so that it is rather the manufacturing engineer’s point of view than the design engineer’s needs.

4. Classification and coding: it is an essential step for full exploitation of the benefits of GT. It is more commonly used in forming part families and machine groups or cells for GT application.

5. Mathematical programming/expert systems: it is the new concept for generative coding and computer-oriented schemes. Expert systems are being studied according to recent developments in the CIM implementation.

**The ‘bottom-up’ approach**

The ‘bottom-up’ approach, as the name suggests, develops the CAPP system by means of tracing the task of process planning from the finished part to the raw material. This is a conventional computer assisted process planning method oriented toward the variant approach. As an example, Fig.2.1 illustrates the schematic of a ‘bottom-up’ CAPP system.

**The ‘top-down’ approach**

The ‘top-down’ approach, contrary to the ‘bottom-up’ approach, develops the CAPP system by means of tracing the task of process planning from top to bottom. This is an automated computer aided process planning method oriented toward the generative approach. In the ‘top-down’ approach the computerized system works in the sequence
determination of overall strategy, analysis of detailed tactics, analysis of the production process for each individual component etc., as previously listed (the other steps of process planning). This requires that, in the first instance, the general rules or techniques of the manufacturing strategy must be built into algorithms which can operate on brief input data describing the geometric features and engineering requirements as they relate to individual
component parts (Marshall 1985). As an example, Fig.2.2 illustrates an architecture of a ‘top-down’ CAPP system.

Since the two methods are not isolated from each other, sometimes a mixture of ‘top-down’ and ‘bottom-up’ methods can produce some good results in enabling inference manufacturing logic to be built into a CAPP system.

Figure 2.2 An architecture of a ‘top-down’ system
AI techniques and expert systems

The term Artificial Intelligence (AI) was created by McCarthy at MIT between the 1950s. General AI methods attempt to solve generalized problems. The CAPP systems in which AI techniques have been applied are called Expert System (ES) or Knowledge Based Systems (KBS). Such an expert system can be defined as a tool which has the capability to understand problem specific knowledge and use the domain knowledge intelligently to suggest alternative paths of action.

At the beginning of the 1980s CAPP has not yet shown many successes in spite of all the effort put into research and development. Some efficient methodologies have to be explored to realize the automatic function of the generative process planning approach. Since process planning in the manufacturing environment is an area requiring a considerable amount of human expertise, KBES are considered to have the largest potential for development of CAPP systems.

A KBES generally consists of four fundamental components:

(1) a knowledge base
(2) a knowledge acquisition mechanism
(3) a recognition/interference mechanism
(4) a user interface scheme
The general structure of such a system is shown in Fig. 2.3. Such a system can be further classified into one of two categories: (1) rule-based systems or (2) frame-based systems, in terms of knowledge representation and inference mechanism.

![Diagram of a general KBES](image)

Figure 2.3 Architecture of a general KBES

1. Rule-based systems, also called the rule-driven approach, make use of an IF<antecedent> THEN<consequent> representation of knowledge. Most rule-based systems consist of two major components (1) a knowledge base which contains the rules and (2) an inference engine which controls the invocation of these rules. A simple example of such an “IF-THEN” rule is:

   IF: the material is hard

   THEN: the drilling feed should be small
2. The 'frame-based' system, also called the pattern-directed approach, is based upon the construction of pyramid-like networks of know-how which is referred to as a knowledge hierarchy. The knowledge base of a frame-based system consists of a knowledge sketch and knowledge details. The knowledge sketch is characterized by three basic primitives: (1) the entities; (2) the attributes; and (3) the relationships. The knowledge details comprise both factual and procedural knowledge which reside in data files and dictionaries of the knowledge base.

2.2 Reviews on fuzzy system

2.2.1 Overview

Born in the United States around 1965, fuzzy set theory has grown to become a major scientific domain. For the past several years, thousands of commercial and industrial fuzzy systems have been successfully developed. The number of industrial and commercial applications worldwide appears likely to increase significantly in the near future.

The most successful domain has been in fuzzy control of various physical or chemical characteristics such as temperature, electric current, flow of liquid/gas, motion of machines, etc. Also, fuzzy systems can be obtained by applying the principles of fuzzy sets and logic to other areas, for example, fuzzy knowledge-based systems such as fuzzy expert systems which may use fuzzy if-then rules; "fuzzy software engineering" which may incorporate fuzziness in programs and data; fuzzy information; fuzzy pattern
recognition which deals with fuzzy visual or audio signals; applications to machine, economics, and management problems which involve fuzzy information processing.

<table>
<thead>
<tr>
<th>Category</th>
<th>Application Area Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>Control is the most widely applied category today.</td>
</tr>
<tr>
<td>Pattern recognition</td>
<td>Image (e.g., OCR), audio, signal processing.</td>
</tr>
<tr>
<td>Quantitative analysis</td>
<td>Operations research, statistics, management.</td>
</tr>
<tr>
<td>Inference</td>
<td>Expert systems for diagnosis, planning, and prediction, etc.</td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>Database</td>
</tr>
</tbody>
</table>

Table 2.1 Generic categories of fuzzy system applications

<table>
<thead>
<tr>
<th>Transportation</th>
<th>(subways, helicopters, elevators, traffic control, and air control for highway tunnels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automobiles</td>
<td>(engines, brakes, transmissions, cruise control systems)</td>
</tr>
<tr>
<td>Consumer electronics</td>
<td>(washing machines, driers, refrigerators, vacuum cleaners, TVs, VCRs, air conditioners, microwave ovens, shower system, video cameras)</td>
</tr>
<tr>
<td>Robotics</td>
<td></td>
</tr>
<tr>
<td>Computers</td>
<td></td>
</tr>
<tr>
<td>Telecommunications</td>
<td></td>
</tr>
<tr>
<td>Other industries</td>
<td>(steel, chemical, power generation, construction, nuclear, aerospace)</td>
</tr>
<tr>
<td>Engineering</td>
<td>(electrical, mechanical, civil, environmental, geophysics)</td>
</tr>
<tr>
<td>Safety / Maintenance</td>
<td></td>
</tr>
<tr>
<td>Agriculture</td>
<td></td>
</tr>
<tr>
<td>Medicine</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>(credit evaluation, risk assessment, stock picking, marketing analysis, production management, scheduling, decision-support systems)</td>
</tr>
<tr>
<td>Education</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2 A partial list of application areas of fuzzy systems
Although there are numerous fuzzy systems in many application types and domains, most of them are based on a relatively simple idea. A fuzzy system allows a gradual and continuous transition, say, from 0 to 1, rather than crisp and abrupt change between binary values of 0 and 1.

These are certain particular characteristics of fuzzy systems that give them better performance for specific applications. In general, fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system with a mathematical model that is difficult to derive. For values of a system may involve fuzziness, be inaccurate, or incomplete. Similarly, the formulas or inference rules to derive conclusions may be incomplete or inaccurate. Fuzzy logic allows decision making with estimated values under incomplete information. Note that the decision may not be correct and can be changed at a later time when additional information is available. Complete lack of information will not support any decision making using any form of logic. For difficult problems, conventional nonfuzzy methods are usually expensive and depend on mathematical approximations (e.g., linearization of nonlinear problems), which may lead to poor performance. Under such circumstances, fuzzy systems often outperform conventional methods such as a proportional, integral, and differential (PID) control.

Fuzzy system approaches also allow us to represent descriptive or qualitative expressions such as “slow” or “moderately fast,” and these are easily incorporated with symbolic statements. These expressions and representations are more natural than mathematical equations for many human judgmental rules and statements.
When fuzzy systems are applied to appropriate problems, particularly the type of problems described previously, their typical characteristics are faster and smoother response than with conventional systems. This translates to efficient and more comfortable operations for such tasks as controlling temperature, cruising speed, for example. Furthermore, this will save energy, reduce maintenance costs, and prolong machine life. In fuzzy systems, describing the control rules is usually simpler and easier, often requiring fewer rules, and thus the systems execute faster than conventional systems. Fuzzy systems often achieve tractability, robustness, and overall low cost. In turn, all these contribute to better performance. In short, conventional methods are good for simpler problems, while fuzzy systems are suitable for complex problems or applications that involve human descriptive or intuitive thinking.

2.2.2 Procedure for developing fuzzy system

The basic steps for developing a fuzzy system are (Munakata, 1994):

1. Determine whether a fuzzy system is a right choice for the problem. If the knowledge about the system behavior is described in approximate form or heuristic rules, then fuzzy is suitable. Fuzzy logic is also useful in understanding and simplifying the processing when the system behavior requires a complicated mathematical model.

2. Identify inputs and outputs and their ranges. Range of sensor measurements typically corresponds to the range of input variable, and the range of control actions provides the range of output variable.
(3) Define a membership function for each input and output parameter. How many membership functions are required is a choice of the developer and depends on the system behavior.

(4) Construct a rule base. It is up to the designer to determine how many rules are necessary and when to stop adding rules.

(5) Verify that the rule base output within its range for some sample inputs, and further validate that this output is correct and proper according to the rule base for the given set of inputs.

2.3 Motivation of the research

Although much effort has been spent on CAPP and fuzzy systems, there are few references related to a fuzzy system for drilling process planning. Some references group drilling planning into process planning of metal cutting approaches, such as turning, milling, etc. There is no detail discussion on CAPP using fuzzy set theory for multiple hole drilling process and that is the reason why so much effort was put into this topic.

Through this research, a “Smart Drilling System” will be presented, which is based on a fuzzy logic algorithm. The system will automatically arrange the process planning of drilling, such as selecting tools, arranging the drilling sequence, etc. This system will also automatically calculate the cycle time and generate the NC code for later simulation. Based on the results of simulation, the optimal process planning of drilling can be derived which will significantly save cutting time and cost.
CHAPTER III

METHODS USED IN THE SYSTEM

This chapter describes the theoretical background of the Smart Drilling System. These include (1) decision logic (2) fuzzy logic and its application in smart drilling system (3) optimization mechanism in drilling (4) cost model of the drilling process.

3.1 Decision Logic

In a generative process planning system, the system decision logic directs the flow of information. The decision logic determines how a process or processes are planned. The major function of the decision logic is to match the process capabilities with the design specification. Process capabilities can be described by “IF ... THEN...” expressions. Such expressions can be translated into logic statements in a computer program.

Several methods can be used to describe the decision structure of process planning. The knowledge representation methods are related directly to the decision logic in these systems. In total, there are three decision logics applied to process planning systems:

1. Decision trees
2. Decision tables
3. Artificial Intelligence
Among the above three alternatives, decision tree and decision tables are used in the smart drilling system.

3.1.1 Decision trees

A decision tree is a graph with a single root and branches emanating from the root. Conditions (IF) are set on the branches of the tree and predetermined actions can be found at the junction of each branch.

A decision tree can be implemented as either computer code, or presented as data. When a decision tree is implemented in computer, the tree can be directly translated into a program flowchart. The root is the start node, and each branch is a decision node. Each branch has a decision statement (a true condition and a false condition). At each junction, an action block is included for the true condition. For a false condition, another branch might be taken or the process might be directed to the end of the logic block. When the false condition includes another branch, these two branches are said to branch from an OR node. When the false condition goes directly to the end of an action block which is rooted from the same decision statement, the current branch and the following branch are part of the same AND node. A decision statement can be a predicate or a mathematical expression.

For example, decision trees are implemented in the smart drilling system to group all holes into three separate groups:

- diameter of holes ≤ d₁, without predrill
- d₁ < diameter of holes ≤ d₂, require at least 1 predrill
- diameter of holes > d₂, require at least 2 predrills
The figure of above decision tree is shown in Figure 3.1

![Decision Tree](image)

Figure 3.1 A decision tree in proposed system

### 3.1.2 Decision Tables

Decision tables used in the proposed drilling system are named as "Tmatrix". An example of Tmatrix is shown in Figure 3.2. The tool matrix is used to express the relations between the operations and the tools, along the horizontal axis the operations and along the vertical axis the cutting tools. Assume that the operation in column j can be performed by using the tool from row i then $T_{matrix}(i,j) = 1$. 

25
### Figure 3.2 Example of Tmatrix

For example, in Figure 3.2, there are four drills that are d1 to d4, eight operations that are to drill hole 1 to hole 8. The drill d1 could finish hole 1, hole 2 and hole 3 (obviously they have the same diameter which equals diameter of tool d1). Hole 4 and hole 5 could be cut with the drill d2. Using tool d3, hole 6 and hole 7 could be drilled. Finally the hole 8 could be finished by using drill d4. Within the smart drilling system, the Tmatrix is arranged in sequence, which is from the smallest to the biggest. For example, diameter of the drills in Figure 3.2, the sequence is d1<d2<d3<d4. Similarly, for the hole sizes, hole 1 is the smallest and hole 8 is the largest.

In this way it is possible to express that different operations can use the same tool and that for one operation more than one tool is suitable (alternative tools). If the tool selection module prefers one tool over another for a given operation, then it should only return its favorite tool and not the less favorable alternatives.
3.2 Fuzzy Logic

3.2.1 Foundations of fuzzy logic

3.2.1.1 Fuzzy sets

Fuzzy logic starts with the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. In an ordinary (nonfuzzy) set, an element of the universe either belongs to or does not belong to the set. That is, the membership of an element is crisp—it is either yes or no, on or off. A fuzzy set is a generalization of an ordinary set by allowing a degree of membership for each element. A membership degree is a real number on [0,1]. In extreme cases, if the degree is 0 the element does not belong to the set, and if 1 the element belongs 100% to the set.

Let fuzzy set A be represented as $A = \{ u / a(u) \mid u \in U \}$, where $u$ is an element, $a(u)$ the membership function to represent the degree, and $U$ the universe. Here “$u \in U$” means “every $u$ in $U$.” Fuzzy operations that have counterparts include the union, intersection, complement, binary relation, and composite of relations. Operations such as fuzzification are unique to fuzzy sets. There are several different ways of defining the union, etc. The most common definitions are:

$$A \cup B = \{ u / \max(a(u), b(u)) \mid u \in U \}$$

$$A \cap B = \{ u / \min(a(u), b(u)) \mid u \in U \}$$

complement $A = \{ u / (1- a(u)) \mid u \in U \}$

Based on these definitions, we can derive the fuzzy versions of familiar properties in ordinary sets, such as commutative laws, deMorgan’s law, etc.
Let A be a fuzzy set of universe U and B a fuzzy set of universe V. The Cartesian product U × V is the one defined for ordinary sets. The Cartesian product A × B is defined by

\[
\{(u,v) / \min(a(u),b(v)) \mid u \in U, v \in V\}
\]

A fuzzy binary relation (or simply fuzzy relation or relation) R from U to V is a fuzzy subset of U × V:

\[
R=\{(u,v) / m(u,v) \mid u \in U, v \in V\}
\]

where m(u,v) is membership function. A fuzzy relation R from A to B can be described as

\[
R=\{(u,v)/m(u,v)lm(u,v) \leq a(u), m(u,v) \leq b(u), u \in U, v \in V\}
\]

Let R be a relation from fuzzy sets A to B, and similarly let S be a relation from B to C. The composition of R and S from A to C denoted R o S is a fuzzy relation defined by:

\[
R \circ S = \bigcup_{A \times C} A \times C \bigcap_B \{\max(a,c) / \min(m_A(a,b), m_B(b,c))\}
\]

Generalizations of these definitions to n-ary are straightforward.

As fuzzy sets are extensions of ordinary sets, fuzzy logic is an extension of ordinary logic. As there are correspondences between ordinary sets and ordinary logic, so are there correspondences between fuzzy set theory and fuzzy logic; for example, union \( \rightarrow \text{OR} \), intersection \( \rightarrow \text{AND} \), and complement \( \rightarrow \text{NOT} \). The degree of an element in a fuzzy set may correspond to the truth value of a proposition in fuzzy logic. Fuzzy logic can represent fuzzy implications such as A \( \Rightarrow \) B, i.e., “if A then B,” where A and B are fuzzy sets. A fuzzy implication is viewed as describing a relation between two fuzzy sets. There are different ways of defining a fuzzy implication as a relation. A common definition is A \( \Rightarrow \) B = A × B, where A × B is the Cartesian product of fuzzy sets A and B.
An extension is "if $A_1$ and $A_2$ then $B$," in which case $A_1$ AND $A_2$ can be substituted in place of $A$.

Let $R$ be a fuzzy relation from $U$ to $V$, $X$ be a fuzzy subset of $U$, $Y$ be a fuzzy subset of $V$, and $Y = X \circ R$. This is called the compositional rule of inference, and $Y$ is said to be induced by $X$ and $R$. *Fuzzy inference* is based on fuzzy implication and the compositional rule of inference. The basic steps are as follows. Suppose that we are given:

1) Implication: "if $A$ then $B$"

2) Premise: " $X$ is true"

3) Conclusion $Y$.

If we want to derive the conclusion, the following steps will be performed:

Step 1: Compute the fuzzy implication as a fuzzy relation $R$, i.e. " if $A$ then $B$":

$R = A \times B$

Step 2: Induce $Y$ by $Y = X \circ R$.

3.2.1.2 *Linguistic variables*

A concept that plays a central role in the applications of fuzzy logic is that of a linguistic variable (Zadeh, L.A., 1973). The concept of a linguistic variable has become sufficiently well understood to make it unnecessary to dwell upon it here. There is, however, one basic aspect of the concept of a linguistic variable which is worthy of note since it is at the heart of its utility.
Specially, consider a linguistic variable such as *drilling feed* whose linguistic values are *small, medium* and *large*, with *small* defined by a membership function such as shown in Figure 3.3.

![Figure 3.3 Linguistic and numerical values of small](image)

Clearly, a numerical value such as 0.01 mm/rev is simpler than the function *small*. But *small* represents a choice of one out of three possible values whereas 0.01 is a choice of one out of, say, 1mm/rev values. The point of this simple example is that the use of linguistic values may be viewed as a form of data compression. It is suggestive to refer to this form of data compression as *granulation*.

The same effect can be achieved, of course, by conventional quantization. But in the case of quantization, the values are intervals whereas in the case of granulation the values are overlapping fuzzy sets. The advantages of granulation over quantization are:

a) it is more general

b) it mimics the way in which humans interpret linguistic values (i.e. as fuzzy sets rather than intervals)
c) the transition from one linguistic value is gradual rather than abrupt, resulting in continuity and robustness.

3.2.1.3 Calculi of fuzzy rules and fuzzy graphs

The concept of linguistic variable serve as a point of departure for other concepts as point of departure for other concepts in fuzzy logic whose use results in data compression. Among these are the concepts of fuzzy if-then rule - or simply fuzzy rule - and fuzzy graph. There is a close relation between these concepts, and both may be interpreted as granular representations of functional dependencies and relations. Viewed from this perspective, fuzzy rules and fuzzy graphs bear the same relation to numerically-valued dependencies that linguistic variables bear to numerically-valued variables.

Like the concept of linguistic variable, the concept of fuzzy rule is sufficiently well understood to make it unnecessary to dwell upon it here. In what follows, we shall confine our attention to the less well-developed concept of a fuzzy graph.

The concept of a fuzzy graph was initially introduced in 1971 (Zadeh, L.A.) and, in a more explicit form, in 1974(Zadeh, L.A.). In an implicit form, the concept of a fuzzy graph underlies the seminal work of Mamdani and Assilian (1975) on fuzzy control. In what follows, we shall assume for notational simplicity that mappings are from \( R \) to \( P \).

As shown in Figure 3.4, a fuzzy graph \( f^* \), of a functional dependence \( f : X \to Y \), where \( X \) and \( Y \) are linguistic variables in \( U \) and \( V \), respectively, serves to provide an approximate, compressed representation of \( f \) in the form

\[
f^* = A_1 \times B_1 + A_2 \times B_2 + \ldots + A_n \times B_n
\]

31
or more compactly,

\[ f^* = \sum_{i=1}^{n} A_i \times B_i \]

![Diagram of function and its fuzzy graph]

Figure 3.4 Representation of a function and its fuzzy graph

where the \( A_i \) and \( B_i \), \( i = 1, \ldots, n \), are contiguous fuzzy subsets of \( U \) and \( V \), respectively; \( A_i \times B_i \) is the cartesian product of \( A_i \) and \( B_i \); and + is the operation of disjunction, which is usually taken to be the union. Expressed more explicitly in terms of membership functions of \( f^* \), \( A_i \) and \( B_i \), we have

\[
\mu_{f^*}(u, v) = V_i(\mu_{A_i}(u) \land \mu_{B_i}(v))
\]

where \( \land = min \)

\( \lor = max \),

32
\[ u \in U \]
\[ v \in V \]

In a general setting, in place of \( \land \) and \( \lor \) we may employ t-norms and s-norms (Zimmerman, H.J., 1990).

Alternatively, a fuzzy graph may be represented as a fuzzy relation \( f^* \)

\[
\begin{array}{c|c|c}
  \text{f}^* & \text{A} & \text{B} \\
  \hline
  A_1 & B_1 \\
  A_2 & B_2 \\
  . & . \\
  . & . \\
  . & . \\
  A_n & B_n \\
\end{array}
\]

Table 3.1 A fuzzy relation \( f^* \)

or a collection of fuzzy if-then rules

\[ f^* \text{ if } X \text{ is } A_1 \text{ then } Y \text{ is } B_1 \]

\[ \text{if } X \text{ is } A_2 \text{ then } Y \text{ is } B_2 \]

\[ \text{. . . . . . } \]

\[ \text{if } X \text{ is } A_n \text{ then } Y \text{ is } B_n \]

with the understanding that the fuzzy if-then rule
if $X$ is $A_i$ then $Y$ is $B_i$, $i = 1, \ldots, n$

is interpreted as the joint constraint on $X$ and $Y$ defined by

$$(X,Y) \text{ is } A_i \times B_i$$

For example, with this understanding the fuzzy rule set

$$f^* \text{ if } X \text{ is small then } Y \text{ is large}$$

$$\text{if } X \text{ is medium then } Y \text{ is medium}$$

$$\ldots \ldots$$

$$\text{if } X \text{ is large then } Y \text{ is small}$$

may be represented equivalently as the fuzzy graph

$$f^* = \text{small } \times \text{large } + \text{medium } \times \text{medium } + \ldots + \text{large } \times \text{small}$$

In effect, a fuzzy graph approximation to a given function combines a relational approximation with data compression (see Figure 3.5)
3.2.1.4 Operations on Fuzzy graphs

A key issue in the calculus of fuzzy graphs relates to the development of computational methods for performing various basic operations on fuzzy graphs. The operations in question are generalizations of the corresponding operations on crisp (non-fuzzy) functions and relations.
In dealing with this issue, it turns out that the necessary computations can be greatly simplified if an operation, $\cdot$, is monotonically nondecreasing, i.e., if $a, b, a, b$, and $b, b$ are real numbers, then

$$a \geq a, b \geq b \rightarrow a \cdot b \geq a \cdot b$$
$$a \leq a, b \leq b \rightarrow a \cdot b \leq a \cdot b$$

For such operations, it can readily be shown that $\cdot$ distributes over $\lor$ (max) and $\land$ (min). Thus

$$a \cdot (b \lor c) = a \cdot b \lor a \cdot c$$
$$a \cdot (b \land c) = a \cdot b \land a \cdot c$$

This implies that if

$$f^* = \sum_i A_i \times B_i$$

is a fuzzy graph and $C$ is a fuzzy set then

$$C \cdot (\sum_i A_i \times B_i) = \sum_i C \cdot (A_i \times B_i)$$

As an illustration, consider the problem of finding the intersection of fuzzy graphs $f^*$ and $g^*$ (Figure 3.6), where

$$f^* = \sum_i A_i \times B_i$$

and

$$g^* = \sum_j C_j \times D_j$$

In this case, we have
\[ f^* \cap g^* = \sum_{i,j} (A_i \times B_j) \cap (C_j \times D_i) \]

which in the view of the distributivity of \( \cap \) reduces to

\[ f^* \cap g^* = \sum_{i,j} (A_i \cap C_j) \times (B_i \times D_j) \]

This result has an immediate application to the interpolation of fuzzy if-then rules, a problem which plays a key role in fuzzy control.

![Figure 3.6 Intersection of fuzzy graphs f* and g*](image)

More specifically, the problem of interpolation may be expressed as an inference query.
\[(X, Y) \text{ is } \sum_i A_i \times B_i\]

\[X \text{ is } A\]

\[Y \text{ is } B\]

On representing the major premise in the query as a fuzzy graph \(f^* = \sum_i A_i \times B_i\) and the minor premise as a cylindrical extension, \(\bar{A}\), of the fuzzy set \(A\) (see Figure 3.7), the computation of \(B\) reduces to that of finding the intersection of \(f^*\) and \(\bar{A}\) and projecting in the resulting fuzzy set on \(V\), the domain of \(Y\).

![Figure 3.7 Intersection of \(f^*\) and \(\bar{A}\)](image)

Thus,
\[ B = \text{proj} V(\bar{A} \cap (\sum_i A_i \times B_i)) \]

which reduces to

\[ B = \text{proj} V(\sum_i (A_i \cap A_i) \times B_i) \]

or, more compactly,

\[ B = \sum m_i \wedge B_i \]

where

\[ m_i = \sup(A \cap A_i) \]

represents the degree of match between \( A \) and \( A_i \) (see Figure 3.8)

![Figure 3.8 The meaning of the degree of match between A and A_i](image)
It should be noted that in a different guise this technique of interpolation was employed in the seminal paper of Mamdai and Assilian (1975) and is currently used in most rule-based control systems.

As a further example, consider the problem of combining \( f^* \) and \( g^* \) through the minimum operator. Thus, if

\[
f^* = \sum_i A_i \times B_i
\]

and

\[
g^* = \sum_i A_i \times C_i
\]

and \( \text{min} \) is the minimum operator, then

\[
f^* \text{min} g^* = \sum_i A_i \times (B_i \text{min} C_i)
\]

where \( B_i \text{min} C_i \) is the minimum of \( B_i \) and \( C_i \) computed through the extension principle. This result makes it possible to compute the intersection, \( F \cap G \), of fuzzy sets \( F \) and \( G \) whose membership functions \( u_F \) and \( u_G \) are represented qualitatively in the form of fuzzy if-then rules (see Figure 3.9).
Figure 3.9 Intersection of fuzzy sets $F$ and $G$ whose membership functions are represented as fuzzy graph

$F$: if $X$ is $A_i$ then $u_F$ is $B_i$

$i=1, \ldots, n$

$G$: if $X$ is $A_i$ then $u_G$ is $C_i$

$i=1, \ldots, n$

An important practical application of the fuzzy graph representation of the intersection $F \cap G$ relates to the case where $F$ and $G$ represent two conflicting goals and $F \cap G$ a maximizing decision (Bellman, R.E. and Zadeh, L.A., 1970).

The concept of a fuzzy graph has an important connection with the representation of fuzzy relations. Thus, if $R$ is a fuzzy relation with attributes which take linguistic values (see Figure 3.10) then $R$ may be represented as a fuzzy graph.
Figure 3.10 A relation with fuzzy-valued attributes

\[
R = R_{11} \times R_{12} \times R_{13} + R_{21} \times R_{22} \times R_{23} + R_{31} \times R_{32} \times R_{33}
\]

The representation of a fuzzy relation as a fuzzy graph may be applied to the representation of diagnostic tableaus (see Figure 3.11) in which the entries are linguistic values of tests and corresponding faults.

Figure 3.11 Diagnostic tableau. L= low; M= medium; H= high; Z= zero

42
Thus, if the result of tests $T_1$, $T_2$, and $T_3$ are $L$, $L$ and $M$, respectively, then the degrees to which the faults $F_1$ and $F_2$ are present are $L$ and $Z$, respectively. If the results of tests are not in the tableau, then the degrees of $F_1$ and $F_2$ may be computed through the use of interpolation applied to the fuzzy graph

$$DT = L \times L \times M \times L \times Z + M \times H \times L \times M \times Z + H \times L \times L \times H \times L$$

3.2.2 Application of fuzzy logic

3.2.2.1 Fuzzy Decision Table (FDT)

A crisp table (two values: 0 & 1) may be extended to include fuzziness in the condition part and/or in the action part, which then gives rise to the notion of a fuzzy decision table (FDT). Fuzziness in the condition part can be expressed by fuzzy conditions (in form of simple predicates) such as "cutting speed is fast", "feed is small", etc. In a FDT, these linguistic terms and fuzzy sets appear with condition states ($S_{ik}$) and/or with action subjects ($A_{ij}$). More formally, a FDT is defined as follows:

**Definition**: Let $CS_i$ be a condition subject with domain $CD_i$ ($i = 1, \ldots, \text{cnum}$), $Ct_i$ be a set of condition states $S_{ik}(k=1,\ldots,n_i, i = 1, \ldots, \text{cnum})$ with $S_{ik}$ being a fuzzy logic expression, $A_{ij}$ be an action subject incorporated with linguistic terms and fuzzy sets, and $AV_j=\{\text{true} (\cdot), \text{false}(-), \text{nil}(\cdot)\}$ be an action value set ($j = 1, \ldots, \text{anum}$), then a fuzzy decision table (FDT) is a function from $CT_1 \times CT_2 \times \ldots \times CT_{\text{cnum}}$ to $AV_1 \times AV_2 \times \ldots \times AV_{\text{anum}}$ such that each possible condition combination is mapped into one action configuration.

Apparently, a crisp DT is a special case of a FDT. Note that in the previous definition and hereafter we assume that any fuzzy set concerned is a normalized fuzzy set,
and that any condition is composed of a simple predicate of such kind as "CS, is A" or 
"CS, is in A".

The construction of FDTs can proceed mainly according to the steps of the crisp case, however, some extensions are needed. For example, extra steps are necessary to 
specify fuzzy sets involved in condition or actions, some provisions are needed to handle 
fuzzy decision rules, etc.

As far as the properties of DTs (completeness, exclusivity, correctness) are 
concerned, it can be seen that definition guarantees the completeness because any 
possible condition combination will lead to a decision in terms of action configurations. 
Moreover, since in general fuzzy sets appear in condition states, the degree of matching 
between a (given) possible condition combination and a FDT is not necessarily 0 or 1 as in 
the crisp case, but a value in [0,1]. Here, when talking about a condition combination 
with fuzziness, we refer to a fuzzy extension of the "AND" operator $^f : [0,1] \times [0,1]$ 
$\rightarrow [0,1]$, such that $^f(0,a) = ^f(a,0) = 0$ and $^f(1,b) = ^f(b,1) = b$. Examples of such $^f$ are 
min(minimum) and * (multiplication).

In addition, the degree of matching between two conditions can be evaluated by a 
closeness measure cm: $F(D) \times F(D) \rightarrow [0,1]$ with cm(A,A) = 1, cm(A,B) = cm(B,A), and 
if supp(A) $\cap$ supp(B) = 0 then cm(A,B) = 0, where A and B are fuzzy sets in $F(D) = \{F | F$ 
is a fuzzy set on $D\}$, and supp(A) and supp(B) are supports of A and B respectively. 
Recall that the support of a fuzzy set is the set of all the elements whose membership 
degrees are greater than zero.

An example of such cm is:
(1) \( \sup \min(A(x), B(x)) \)

More formally, the concept of completeness can be defined in the context of FDTs. Let FDT be a fuzzy decision table. The condition states \( S_{ik} \) (\( k = 1, \ldots, n_i \)) of a CT\(_i\) are called complete if and only if the union of all the supports of fuzzy sets involved in all \( S_{ik} \) covers the condition domain CD\(_i\). A FDT is called complete if and only if the condition states of each CT\(_i\) (\( i = 1, \ldots, \text{cnum} \)) are complete.

Thus the completeness of a FDT can be guaranteed if there exists at least one column in the FDT with which the degree of matching a given condition combination is greater than zero. This is shown in the following theorem.

**Theorem 1.** A FDT is complete if for any given condition combination, there exists at least one column in the FDT with which the degree of matching is greater than zero.

**Proof:** Suppose the FDT is not complete, then according to definition 3 there exists an element \( e \) in CD\(_i\) such that \( e \) does not belong to the union of all the supports of fuzzy sets involved in \( S_{ik} (k=1, \ldots, n_i) \). Now let a condition combination is merely composed of “CS\(_i\) is \{1/e\}”, then the degree of closeness between \{1/e\} and any other fuzzy set \( F \) in any \( S_{ik} \) of the FDT is zero because \( e \) has a zero membership degree in \( F \). However, this is a contradiction to the fact that the degree of matching between any condition combination and the FDT is greater than zero.
The notion of correctness can be determined in a similar way to that of the crisp case. That is, it can be checked by the designer whether the FDT reflects what was meant by the user.

3.2.2.2 Fuzzy decision making using FDT

Fuzzy decision making is to allow fuzzy consultation of decision tables. On one hand, fuzzy consultation can be made on crisp decision tables. This is of a great value because existing (crisp) DTs can then be utilized. On the other hand, fuzzy consultation can generally be made on fuzzy decision tables. In either case, however, a decision or action configuration cannot be taken by merely checking with each column of the table to match (perfectly) a given condition configuration. Instead, the degree of matching between the given condition combination and each column should be evaluated. As a result, more than one action configuration may be chosen, each with a degree in $[0,1]$.

There exist various ways to derive the degree ($\alpha_i$) associated with an action configuration. One way is shown in (2):

\[ \alpha_i = cm(c_1, F(S_{1k})) \land cm(c_2, F(S_{2k})) \land \cdots \land cm(c_{num}, F(S_{num \cdot k})) \]

where $c = \{ c_1, c_2, \ldots, c_{num} \}$ is a given condition combination with each $c_i$ being the fuzzy set involved, $F(S_{ik})$ is a fuzzy set involved in $S_{ik}$ appearing at column $i$, $cm$ is closeness measure, and $\land$ is a fuzzy "AND" operator as described previously. This setting says that the higher the degree of matching between condition combinations is, the higher the degree with which the corresponding action configuration is associated. This is intuitive appealing. In fact, these two degrees are set equal in (2).
It is worthwhile to emphasize that the above mentioned process of fuzzy decision making is to choose those actions that are tabulated in a FDT, based upon the matching of fuzzy conditions. In general, however, fuzzy decision making can be dealt with under the concept of approximate reasoning, since a column of a decision table may be viewed as an if-then rule. Thus, fuzzy implication operators will play an important role. Consider the generalized modus ponens (Kerre E. E., 1994):

if \( X \) is \( A \) then \( Y \) is \( B \)

\[ X \text{ is } A' \]

\[ \sup_{x \in D_1 \times \ldots \times D_m} T(A'(x), B_j(y)) \]

where \( A, B, A', \) and \( B' \) are fuzzy sets. In the context of a FDT, "\( X \) is \( A \)" and "\( Y \) is \( B \)" are expressed in the condition part and action part respectively, "\( X \) is \( A \)" is a given condition, and "\( Y \) is \( B \)" is an action to take. Note here that \( A' \) and \( B' \) are generally different from \( A \) and \( B \). As a matter of fact, "\( Y \) is \( B \)" is a new piece of information, knowledge or action that is derived from the FDT. In a FDT, fuzziness in the action part is modeled with action subjects in terms of linguistic terms and fuzzy sets \( (B_j) \). Suppose \( B_j \) is in action subject \( A_{S_j} \), \( B_j \) is determined depending on the given condition combination, \( B_j \), and the corresponding column, using a form of so-called T-norm (e.g. \( \wedge f \)) and a fuzzy implication operator \( I \). Concretely, let \( B_j \) and \( B_j' \) be fuzzy sets on \( D_j \), \( j=1, \ldots, \mathrm{cnum} \), for any \( y \) in \( D_j \),

\[ (3) \quad B_j'(y) = \sup_{x \in D_1 \times \ldots \times D_m} T(A'(x), B_j(y)) \]
where I is a fuzzy implication operator, T is a T-norm (e.g. min, *, W,Z), A(x) = c(x) / \wedge \text{f c}(x_{cnum}) is the degree of vector x corresponding to the given condition combination with each \text{f c_i} being a fuzzy set involved for CS_i, and A(x) = F(S_{ik})(x_{i}) \wedge \text{f F(S_{ik2})(x_{i})} \wedge \text{f F(S_{iknum})(x_{i})} is the degree of x concerned, with each F(S_{ik}) being a fuzzy set involved in F(S_{ik}) appearing at the column. Usually, a specific I or T can be chosen considering certain intuitive knowledge. For instance, we would reasonably expect to have B_{j'} = B_{j} when A' = A. This can be satisfied with T=min and I=I_{g}(i.e., I_{g}(a,b) = 1 for a \leq b; I_{g}(a,b) = b for a>b).

When consulting a FDT with fuzziness, an action configuration may be chosen with or without involving computing B_{j'}. If an action subject A_{s_j} does not involve any fuzzy set, then the action chosen is A_{s_j} itself, otherwise, B_{j'} will be derived. For example, if an action subject involves a fuzzy set B_{j}, e.g., "increase feed to medium (B_{j})", then the action configuration will contain an action, e.g., "increase feed to B_{j}'. The degree associated with each action configuration may also be determined using (2). Other alternatives are possible, for example, I(cm(A,A'), cm(B,B'))), but need to be further explored.

3.2.3 Fuzzy Decision Table in smart drilling system

It is obvious that all the possible values for previous decision table Tmatrix are either 1 or 0. For direct drilling, the Tmatrix works fine, because there could be only one choice for the drill which is exactly the same as the diameter of the hole. However, if a bigger size hole requires several predrills and there also exists some alternative drills
available, it will be not suitable to use 1 or 0 to represent the situations. Hence, fuzzy decision table was introduced in solving the problem.

The procedure of generating the fuzzy decision table contains the following steps:

- Read in hole data and search holes with bigger size diameter
- Find the finish drill which will be used finally, set T(i,j) = 1;
- Search all the drills which are smaller than the finish drill.
- Fuzzify a T(k,j) (where k = 1,2,...,i-1) with the following formula

\[ T(k,j) = \frac{U}{1 + B(D_i - D_k)} \]

where:

\[ U = 1 \text{ if the current drill has been used before.} \]
\[ U = 0 \text{ if the current drill has not been used} \]
\[ B \text{ is a weight parameter, it will concern the following drilling situations:} \]

- depth of holes
- shape of holes
- situation of drills (such as cutter shape, number of cutters)

\[ D_i \text{ is the diameter of finish drill} \]
\[ D_k \text{ is the diameter of current drill} \]

- Search all the values of T(k,j) and find the biggest one, suppose it is T(n,j) then the drill #n will be selected as the predrill.
- Print out the search results.
An example of using fuzzy decision table to select tools is shown in the following, Figure 3.12.

<table>
<thead>
<tr>
<th>Tool</th>
<th>0.2”</th>
<th>0.2”</th>
<th>0.4”</th>
<th>0.4”</th>
<th>0.6”</th>
<th>0.6”</th>
<th>0.8”</th>
<th>1.2”</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2”</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>.4”</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.56</td>
</tr>
<tr>
<td>.6”</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0.63</td>
</tr>
<tr>
<td>.8”</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>0.72</td>
</tr>
<tr>
<td>1”</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3.12  An example of using fuzzy decision table

Based on the values shown in the above table, for the multiple drilling hole, with diameter d = 1.2”, the predrill will be the drill with d = 0.8”, which is obvious that it has the biggest value of 0.72. As to other drills, the fuzzy values are not bigger than 0.72, so they will not be selected as predrill.

Following is the proof of the optimization of this fuzzy algorithm.

- **Complete.** Because all the cutting tools are stored in the decision table. The drills selected in the drilling operation will all be included in the tables, so the decision table contains all possible optimal drills. If there is any optimal drill not included in the decision
table, then it must not be a standard drill. Hence, all the drills discussed here are all standard.

• **Optimal.** All possible predrills are assigned a fuzzy value, $T[i,j]$, which is considered as the degree of being the most suitable predrill. The bigger value, the more suitable the drill will be. Hence, the biggest one will be chosen, then it could guarantee that the most suitable predrill will be selected.

### 3.3 Optimization Mechanism

As discussed before, the major objective of the system is to save cutting time. Then the operation of drilling can be optimized to become more efficient. Within the following, the details of the optimization mechanism are discussed. Optimization in the smart drilling system includes (1) the tool change and (2) the tool travel.

#### 3.3.1 Optimization of tool changes

Only large size holes that require predrills will be considered in the optimization of tool changes. As to smaller holes, the size of drill must match the diameter of hole and hence, is determined. For larger holes, only the finish drills are determined. However, there could be several alternatives for predrills.

The reason why predrilling is required is because of the limitation on the Material Removal Rate (MRR). If the MRR is too large, then the drill would break or drift. In drilling operation, MRR could be calculated as the following:

\[
MRR = \frac{\pi D^2}{4} fN
\]  

(3.1)

Where:
D - diameter of drill

f - feed rate

N - the rpm of the drill

In the proposed system, a hole with diameter bigger than \( D_{\text{max}} \) will require predrilling, the particular value is based on the calculation in terms of MRR. Then maximum drill diameter without predrilling is \( D_{\text{max}} \), or the maximum MRR is \( \text{MRR}_{\text{max}} = \frac{\pi D_{\text{max}}^2}{4} fN \).

Based on the \( \text{MRR}_{\text{max}} \), we could give out the alternative drills for a large size hole. For example, shown in Figure 3.13, there is hole with diameter \( D \), and \( d \) is the unknown diameter of predrill tool. Then \( d \) would satisfy the following equations:

\[
\frac{\pi}{4} fN (D^2 - d^2) < \text{MRR}_{\text{max}}
\]  (3.2)

\[
\frac{\pi}{4} fN (d^2) < \text{MRR}_{\text{max}}
\]  (3.3)

![Figure 3.13 Calculation of alternative drill diameter](image)

Figure 3.13 Calculation of alternative drill diameter
From the above equation, the d could be obtained as a range, suppose they are d1, ..., d4, in which d1 < d < d4. We could find alternative predrill tools inside the Tool matrix as shown in Figure 3.14. It is very easy to see that all the tools between d1 and d4 could be an alternative drill for predrilling the hole with diameter D. Further steps will focus on finding the optimal one and eliminating all the other alternatives. For this procedure, an algorithm will be applied to make sure that the optimal drill is the frequently used one. Then the planning could reduce the number of tool changes.

<table>
<thead>
<tr>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>hole 1</td>
</tr>
<tr>
<td>d1</td>
</tr>
<tr>
<td>d2</td>
</tr>
<tr>
<td>d3</td>
</tr>
<tr>
<td>d4</td>
</tr>
<tr>
<td>d5</td>
</tr>
<tr>
<td>d6</td>
</tr>
</tbody>
</table>

Figure 3.14 An example of Tmatrix contains multiple predrills

As to much larger size holes, which require multiple predrilling, the problem of finding predrills will be complex. For example, there is a larger size hole D, which requires
predrilling two times, shown in Figure 3.15. To handle this problem, the following procedure was adopted in the Smart Drilling System:

![Diagram showing predrilling process](image)

Figure 3.15 An example of determination of multiple predrills

1. Using the following formula to determine diameter of the first predrill $d_1$

   \[
   \frac{\pi}{4} fN (d_1^2) \leq \text{MRR}_{\text{max}}
   \]

   (3.4)

   There will be several choices for the first predrill, then some rules will be applied to find the most desirable one:

   Rule 1: only the drills being used frequently

   Rule 2: only the drills whose diameter is closest to $D_{\text{max}}$

2. For the second predrill, the diameter $d_2$ could be determined by the following:
\[ \frac{\pi}{4} fN \left( D^2 - d_{2}^2 \right) < MRR_{\text{max}} \]  \hspace{1cm} (3.5) \\
\[ \frac{\pi}{4} fN \left( d_{2}^2 - d_{1}^2 \right) < MRR_{\text{max}} \]  \hspace{1cm} (3.6)

The above two rules will also be applied so that the hole could be finished using drills that are already being used for other operations. Then, once the drill (for example, \( d_1 \) or \( d_2 \)) is selected, the predrilling of hole with diameter \( D \) will be finished. Always follow the above procedure to determine the multiple predrills, then the number of tool changes will definitely be reduced.

Finally, for the shape drill, it is easy to figure out the solution as follows:

1. Search all the holes to find holes with exactly the same diameter and shape;

2. Use the same shape drill to finish them all.

All the shape drills are standard in terms of diameter and shape. Once a shape drill is selected, the Smart Drilling System will arrange all the same holes (diameter and shape) to be drilled using this tool.

3.3.2 Optimization of tool traveling

3.3.2.1 Traveling salesman problem in drilling

The most prominent member of the rich set of combinatorial optimization problems is the traveling salesman problem (TSP), the task of finding a route through a given set of cities with shortest possible length.

The drilling problem is a standard application of the symmetric traveling salesman problem. Holes inside the part have to be drilled. The holes may be of different diameters,
depth, and shapes. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool box and change the drilling equipment. This is quite time consuming. Thus it is clear at the outset that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter etc.

Thus, this drilling problem can be viewed as a sequence of symmetric traveling salesman problems, one for each diameter respective drill, where the “cities” are the initial position and the set of all holes that can be drilled with one and the same drill. The “distance” between two cities is the time it takes to move the head from one position to the other. The goal is to minimize the travel time for the head of the machine.

In general, holes are of different diameters. Hence drills have to be changed. In our application drills cannot be changed “on the fly”. The head containing the drill has to move to an origin where the drill can be changed. Since time to change drills is considerable it only makes sense to drill all holes of one diameter before changing the drill.

The problem to be solved here is here to find a sequence in which the holes are to be drilled such that the total production time is minimized. This problem decomposes into a set of single problems for each diameter.

We will now consider such a subproblem. Since the corresponding drill has to be loaded the machine has to move to the origin. Then all holes of the chosen diameter are drilled and the machine returns to the origin to start the next drilling sequence. The time to drill the holes cannot be influenced. Production time can only be decreased if positioning time, i.e., total time needed to reach the positions of the holes, is reduced. This amounts to
solving a symmetric traveling salesman problem in the complete graph on \( n+1 \) nodes, if \( n \) holes are to be drilled (the node \( n+1 \) represents the origin).

The distance between two nodes \( i \) and \( j \) corresponds to the positioning time the machine needs to move from position \( i \) to position \( j \), and therefore depends strongly on the machine characteristics. In practice, usually, this positioning time cannot be computed exactly. Positioning consists of three phases: accelerating the machine, running at full speed, slowing down to a complete stop. For small distances, full speed may not be reached and we may have anomalies in the sense that a farther position can be reached faster than a nearer position. Even if a timing function is available it may be not accurate or so complicated that its evaluation takes too long for large problem instances (where we cannot store a distance matrix). Therefore one has to be satisfied with making reasonable approximations on the true movement time.

3.3.2.2 An algorithm for TSP : Median Finding

Suppose we have a set \( A=\{a_1, a_2, \ldots, a_n\} \). First identify a median of \( A \), i.e., a value \( \bar{a} \) such that half of the elements of \( A \) are below \( \bar{a} \) and half of the elements are above \( \bar{a} \). More precisely, we identify a number \( \bar{a} \) such that we can partition \( A \) into two sets \( A_1 \) and \( A_2 \) satisfying \( A_1 \subseteq \{a_i | a_i \leq \bar{a}\} \), \( A_2 \subseteq \{a_i | a_i \geq \bar{a}\} \), \( |A_1| = \left\lfloor \frac{n}{2} \right\rfloor \), and \( |A_2| = \left\lceil \frac{n}{2} \right\rceil \).

We then sort \( A_1 \) and \( A_2 \) separately. The concatenation of the respective sorted sequences give a sorting of \( A \).

In particular for geometric problems defined on points in the plane we often need to compute horizontal or vertical lines separating the point set into two (approximately...
equally sized) parts. For this we need medians with respect to the x- or y-coordinates of the points.

A natural way to find a median is to sort the \( n \) points. The element at position \( \left\lfloor \frac{n}{2} \right\rfloor \) gives a median. However, one can do better as is shown in the following sophisticated algorithm which is also very instructive. The algorithm requires a subroutine that for a given input \( b \) rearranges an array such that in the first part all elements are at most as large as \( b \) while in the second part all elements are at least as large as \( b \). Assume again that array \( B \) contains \( n \) numbers in \( B[1] \) through \( B[n] \).

**function partition \((B,b)\)**

(1) Set \( i=1 \) and \( j=n \).

(2) Repeat the following steps until \( i \geq j \).

(2.1) Decrement \( j \) by 1 until \( B[j] \leq b \).

(2.2) Increment \( i \) by 1 until \( B[i] \geq b \).

(2.3) If \( i < j \) exchange \( B[i] \) and \( B[j] \).

(3) Return \( j \).

**end of partition**

After execution of this algorithm we have \( B[i] \leq b \) for all \( i=1,2,\ldots,i \) and \( B[i] \geq b \) for all \( i=j,j+1,\ldots,n \).

The procedure for finding a median can now be given. In fact, the procedure does a little bit more. It finds the \( i \)-th smallest element of a set.

58
Procedure \texttt{find\_ith\_element} (B,i)

(1) Partition B into \( \left\lfloor \frac{n}{5} \right\rfloor \) groups of 5 elements each and one last group containing the remaining elements.

(2) Sort each set to find its "middle" element. If the last set has even cardinality \( l \) we take the element at position \( l/2 + 1 \).

(3) Apply \texttt{find\_ith\_element} to find the median \( b \) of the set of medians found in Step(2).

(4) Let \( k = \text{partition} \) (B,b).

(5) If \( i \leq k \) then find the \( i \)-th smallest element of the lower side of the partition. Otherwise find the \( (i-k) \)-th smallest element of the higher side of the partition.

\textbf{End of find\_ith\_element}

The call \texttt{find\_ith\_element} (B, \( \left\lfloor \frac{n}{2} \right\rfloor \)) now determines a median of B. A running time analysis shows that this algorithm runs in linear which is best possible since every element of B has to be considered in a median finding procedure. Hence medians can be found in time \( \Theta(n) \).

\textbf{3.3.2.3 An example of TSP in drilling}

For example, we have five holes with the same diameter D, their hole ID is 1 to 5, shown in Figure 3.16. In the smart drilling system, the drilling sequence is determined as follows:
1. First of all, search all the five holes and find the one that is nearest to point \((0,0,Z)\); from the figure, it is hole 3. This hole will be drilled first.

2. Search all the remaining holes, find the one that is nearest to hole 3, which is hole 1.

3. Repeat the procedure 2 until all the five holes' drilling sequence has been arranged.

Finally the drilling sequence is determined as hole 3 -> hole 1 -> hole 4 -> hole 2 -> hole 5.

![Diagram showing drilling sequence](image)

**Figure 3.16** determination of drilling sequence

### 3.4 Cost model of the drilling process
To solve the optimization problem further, a cost model has to be defined to approximate the real cost. Total costs are calculated by an objective function that combines the total time to manufacture the considered workpiece and resulting manufacturing cost linearly. To keep runtime performance of the optimization method within acceptable limits, the cost model derived should be as simple as possible.

The process model used for the optimization method discussed in this proposal will be described in the following section.

**Determination of the total time to drill the workpiece**

The total time to drill a workpiece is defined by

\[ TM = t_{MD} + t_{MV} + t_{TC} \]  \hspace{1cm} (3.7)

where \( t_{MD} \): total machining time for drilling

\( t_{MV} \): total time for tool movements

\( t_{TC} \): total time for tool exchange

**Machining time for drilling**

The machining time \( t_{MD} \) needed to drill an objective \( O_i \) with the tool \( T_j \) is calculated as follows:

\[ t_{MD}(O_i, T_j) = 2\left[ \frac{l_{\text{max}}}{V_f} \sum_{k=1}^{I} k \right] + \frac{2l}{V_f} \]  \hspace{1cm} (3.8)

where \( l_{\text{max}} \): maximum drilling depth
l : depth of hole to be drilled
k : index for number of drilling cycles
V_{f} : feed rate

**Time for tool movements**

$t_{MV}^{i}$ is defined as time needed within the ith tool to move a tool from the end position of the last drilled hole to the new drilling hole. Look for the shortest path in the sense of shortest time distance where all objects $O_i$ (holes to be drilled) of the work piece are visited exactly once (typical traveling salesman problem):

$$t_{MV} = \min \sum_{i=1}^{n-1} t_{MV}^{i} = \min \sum t_{dist}(\phi(i), \phi(i + 1))$$  
(3.8)

where 
i : index for tool

t_{dist}( ) : function to determine shortest path

$\phi(i)$ : end position of ith tool cycle

$\phi(i + 1)$ : start position of (i+1)-th tool cycle

**Tool exchange time**

The tool exchange time $t_{TCE}$ is the average time period for changing a tool.

$$t_{TC} = n_{TC} \cdot t_{TCE}$$  
(3.9)

where 
$n_{TC} : number of tool changes$

$t_{TC} : total tool change time$
\( t_{TC0} \): per tool change time

**Cost calculation**

Manufacturing cost of a workpiece is composed of machine tool cost and tool cost.

\[
C_M = C_{MT}(t_{MD} + t_{MV} + t_{TC}) + \sum_i C_i^t \frac{t_{Mi}}{T_i}
\]  
(3.10)

where  
\( C_M \): manufacturing cost

\( C_{MT} \): unit production cost

\( C_i^t \): tool cost of \( i \)th tool

\( t_{Mi} \): machining time of \( i \)th tool

\( T_i \): tool life span of \( i \)th tool

The total cost \( C_{Tot} \) is composed of total manufacturing time and manufacturing cost which are being mutually weighted.

\[
C_{Tot} = \gamma \cdot TM + C_M
\]  
(3.11)

where  
\( \gamma \): weighting factor [price unit per time unit]

\( TM \): total manufacturing time

For large values of \( \gamma \) those process variants that are close to the minimum of the total manufacturing time also represent solutions close to the global optimum. For small values of \( \gamma \) the effect is reversed.
CHAPTER IV

SMART DRILLING SYSTEM DESCRIPTION AND
SIMULATION EXAMPLES

4.1 Description of Problem

In this chapter, the use of the system is demonstrated by examples. For multiple hole drilling, in addition to different diameters, the holes may be in different shapes as well, as illustrated Figure 4.1. The other important attributes of the holes include the depth of the holes as well as the locations. In summary, holes can be described as follows:

- diameter of holes
- depth of holes
- shape of holes
- location of holes

![Diagram of different types of holes](Image)

**Countersinking hole**  **Counterboring hole**  **Straight hole**

Figure 4.1 Different type of holes
In C++ language, the above information can be described as a structure named “hole_rec” as shown below:

```c
typedef struct hole_rec {
    int hole_ID;
    char shape;
    float coordi;
    float diam;
    float depth;
} hole
```

For example, 6, countersink, 5.2, 3.4, 1.25, 4.5 represents hole No.6 with countersink shape, and is located at x = 5.2”, y = 3.4”. The diameter of the hole is 1.25” and depth is 4.5” (Assuming that English unit is used).

The Smart Drilling System can design a drilling sequence that minimizes the number of tool changes while ensuring all the holes are drilled properly.

### 4.2 Description of the procedure

The procedure of the system could be summarized and shown in Figure 4.2. The procedure of smart drilling is as follows:

- read hole data
- sort the hole data and list the sorting results
- search and plan drilling sequence
- output the planning sequence
The details of the sorting and searching procedure are displayed in Fig. 4.3 and 4.4.

Figure 4.2 Description of the procedure of smart drilling
Figure 4.3 Details of sorting & listing module

Figure 4.4 Details of searching & matching module
The procedure starts with reading the hole data, following by sorting the diameter of holes and dividing into direct drilling or multiple drilling. The next step is to search the maximum depth with the same diameter. Then the searching results are listed. The drilling process planning is generated according to the following rules:

1. If the hole and drill are the same diameter, then use the drill to finish it.
2. If the hole is a predrill hole and the difference between the hole and the drill is within the range (the value is 0.1"), then use the drill to finish it.
3. Use the drill to finish all possible holes in order to obtain the minimum tool changes.
4. Straight drilling first and then shape drilling to finish.
5. Arrange drills from the smallest to the biggest.

As to the drilling sequence, the Smart Drilling System always follows the drilling sequence, that is, center drilling -> direct drilling -> shape drilling. Center drilling is done to assure that the subsequent drilling operation gets off to a good start. It also helps to locate the drilled hole position and increase the stability of the drilling process.

In the following, an example is used to describe details of the procedure for drilling process planning. To simplify the problem, only three hole records are utilized and they are:

Hole 1, counterboring, 2.5, 2.5, 0.6875, 1.5
Hole 2, counterboring, 4.5, 4.5, 0.6875, 2
Hole 3, countersink, 1.5, 1.5, 1.25, 2.5

The procedure is determined as follows:
1. Search all holes first and find out:
   - hole 1 and hole 2 do not require predrilling
   - hole 3 requires one predrilling

2. Search hole 1 and hole 2 and find that they are the same diameter, then compare
   the depth of the two holes to obtain that direct drill #1 should be a diameter of 0.6875 inch
   and a length of 2 inches.

3. Search hole 3 and determine the diameter of predrill should be 0.6875 inch,
   which equals the diameter of direct drill #1. Then compare the depth of hole 3 with the
   length of direct drill #1 and determine the new direct drill #1 should have a length of 2.5
   inches.

4. Then the process planning for the three holes is:
   - Using the center drill to locate three holes’ positions;
   - Using the direct drill #1 (diam. = 0.6875”; length = 2.5”) to drill all
     three holes;
   - Using the direct drill #2 (diam. = 1.25”; length = 2.5”) to drill hole 3;
   - Using the shape drill (diam. = 0.6875”; counterboring) to finish hole 1
     and hole 2;
   - Using the shape drill (diam. = 1.25”; countersink) to finish hole 3;

In total, to finish the three holes, Smart Drilling selects five drills: 1 center drill, 2
direct drills and 2 shape drills. If using manual drilling (which means to finish the holes
one by one and never group them together), it will have the maximum tool changes. For
this example, manual drilling will require eight tools to finish three holes:
1. Using center drill to finish 3 direct holes;
2. Using direct drill #1 (diam. = 0.6875", length = 1.5") to drill hole 1;
3. Using shape drill #1 (diam. = 0.6875", counterboring) to finish hole 1;
4. Using direct drill #2 (diam. = 0.6875", length = 2") to drill hole 2;
5. Using shape drill #2 (diam. = 0.6875", counterboring) to finish hole 2;
6. Using direct drill #3 (diam. = 0.625", length = 2.5") to predrill hole 3;
7. Using direct drill #4 (diam. = 1.25", length = 2.5") to drill hole 3;
8. Using shape drill #3 (diam. = 1.25", countersink) to finish hole 3.

4.3 The Decision Mechanism

The Smart Drilling System uses a combination of the decision table and decision tree. The data of holes and drills are stored using arrays. When the program performs search, it will go through the relative database and will follow some rule-based algorithm to perform the search.

For example, refer to Figure 3.1, holes are classified into three different groups, which are diam.\( \leq d_1 \), \( d_1 < \text{diam.} \leq d_2 \) and diam.\( > d_2 \), and the statements are the following:

\[
\begin{align*}
\text{IF} & \quad (\text{diam.} \leq d_1) \\
\text{THEN} & \quad (\text{group into type 1})
\end{align*}
\]
{IF
  (d₁<diam<=d₂)
THEN
  (group into type 2)}
{IF
  (diam>d₂)
THEN
  (group into type 3)
END IF}

4.4 System Description

The structure of the Smart Drilling System is shown in Fig. 4.5. The basic step of
the system is to acquire the best knowledge of drilling from various experimental data and
from handbooks or research work. This accumulated knowledge is structured in four
modules: (1) cutting tools module (2) machining data module (3)NC machining module
and (4) Simulation module
4.4.1 Tool selection module

The tool selection module is used for drill selection which is the most critical and time consuming factor of the process. It has a great influence on production cost and productivity.

For tool selection, the input required by the system is the following data: diameter, depth and shape of the holes. The system will then automatically search and evaluate the optimum tool size. For example, if hole 1 to hole 5 have the same diameter 1", but they have different depths, then the optimum tool will be the one with the greatest depth, which could fulfill the requirement of finishing all the five holes.
4.4.2 Cutting condition design module

The generative approach is incorporated for computing and optimizing the machining parameters. In this system optimization is considered based on two objectives:

(1) maximum production rate

(2) minimum cost of production

These objectives are functions of the cutting speed which will produce the above objectives of the machining operation. The generative approach does not rely on handbook recommendations for the selection of cutting speed other than feed and depth of cut, because cutting speed has the greatest effect on tool life. This method uses workpiece, and related Taylor’s constant (C and n) as inputs to compute the optimum cutting conditions.

4.4.3 NC code generation module

This module will automatically generate the NC code (which is APT code) based on the information obtained from the previous process. The following information is required:

- the data of holes (hole id, diameter, shape, coordinates, depth)

- the data for drills (tool id, diameter, shape, length)

- data for cutting condition (speed, feed rate, spindle)

An example of NC code that this system generated, shown in Figure 4.6.
VERICUT-apt
version-2.0
BEGIN
CUTTER/1.30,...,2
SPINDLE/2000
SPINDLE/ON,CCLW
RAPID
GOTO/-.1,-.1,3
RAPID
GOTO/1,1,3
FEDRAT/20,IPM
GOTO/1,1,2,95
GOTO/1,1,2,9
GOTO/1,1,2,85
GOTO/1,1,2,8
GOTO/1,1,2,75
GOTO/1,1,2,7
GOTO/1,1,2,65
GOTO/1,1,2,6
GOTO/1,1,2,55
GOTO/1,1,2,5
PAPID
GOTO/1,1,3
RAPID
GOTO/2,2,3
FEDRAT/20,IPM
GOTO/2,2,2,95
GOTO/2,2,2,9
GOTO/2,2,2,85
GOTO/2,2,2,8
GOTO/2,2,2,75
GOTO/2,2,2,7
GOTO/2,2,2,65
GOTO/2,2,2,6
GOTO/2,2,2,55
GOTO/2,2,2,5

Figure 4.6 an example of generated APT code
4.4.4 Simulation module

After generating the NC code, the smart drilling system will automatically generate a data file that will store the NC code. The simulation was run in VERICUT® system. A user could directly open the NC data file by setting up the file path, then run VERICUT® to simulate the drilling process. VERICUT® is very good software, popular in the tool and die industry. It can simulate the NC cutting using any NC code (e.g. G code, N code and APT code, etc.), which presents a virtual reality of the machining and could be used to check the NC programming. Simultaneously, the result of process planning of the drilling sequence could also be justified. If there are some errors, the system will show some messages and identify the errors automatically.

The smart drilling system can handle the problem of the drilling process and could also be developed to integrate other machining methods, such as milling, turning, etc. Further work is required to fulfill more function requirement for practical machining.

4.5 Simulation Examples

Currently, most of the system implementation has been done and a number of simulations has been conducted. During the simulation, two drilling process plans are compared:

1. **smart drilling**, which utilizes the smart drilling system to automatically generate the process planning for drilling holes.

2. **manual (sequence) drilling**, which does not consider the smart drilling and only finishes the holes one by one without optimal planning.
The input includes data of the holes. Data type is already defined. Each hole is a single record that consists of the following data:

- Hole I.D.
- Hole shape
- Hole coordinates
- Hole diameter
- Hole depth

All the above data value are assigned to arrays, so each record is a group of arrays. The hole data are randomly generated but with some constraints:

a) the range of diameter is from 0.125” to 2.5". This is the normal range of holes in shop floor. Irregular holes, such as large size or mini size holes are not considered.

b) all the diameters are standard, which means that the holes can be finished by standard drills.

A total of 20 simulations was conducted, in which the number of holes ranged from 10 to 40. Result are shown in Table 1, and Figure 4.7(a-e).
<table>
<thead>
<tr>
<th>Num. of holes</th>
<th>Manual</th>
<th>Drilling</th>
<th></th>
<th>Smart</th>
<th>Drilling</th>
<th></th>
<th>Total</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>num. of tool changes</td>
<td>moving distance (mm)</td>
<td>cycle time (sec.)</td>
<td>num. of tool changes</td>
<td>moving distance</td>
<td>cycle time (sec.)</td>
<td></td>
<td>Saving</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>447.111</td>
<td>14.702</td>
<td>11</td>
<td>290.156</td>
<td>7.974</td>
<td>45.8%</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>509.389</td>
<td>16.132</td>
<td>11</td>
<td>365.873</td>
<td>9.826</td>
<td>39.1%</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>754.86</td>
<td>24.975</td>
<td>11</td>
<td>443.81</td>
<td>12.233</td>
<td>51%</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>776.39</td>
<td>25.5</td>
<td>13</td>
<td>486.447</td>
<td>12.93</td>
<td>49.3%</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>903.52</td>
<td>29.37</td>
<td>13</td>
<td>512.86</td>
<td>14.19</td>
<td>51.7%</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>1020.8</td>
<td>34.29</td>
<td>15</td>
<td>657.94</td>
<td>18.33</td>
<td>46.5%</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>29</td>
<td>1016.72</td>
<td>32.994</td>
<td>15</td>
<td>645.331</td>
<td>17.45</td>
<td>47.1%</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>30</td>
<td>1076.31</td>
<td>34.58</td>
<td>15</td>
<td>646.99</td>
<td>17.09</td>
<td>50.6%</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>32</td>
<td>1103.59</td>
<td>35.19</td>
<td>15</td>
<td>555.87</td>
<td>15.18</td>
<td>57%</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>34</td>
<td>1267.89</td>
<td>40.33</td>
<td>15</td>
<td>755.39</td>
<td>19.61</td>
<td>51.8%</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>37</td>
<td>1258.76</td>
<td>40.04</td>
<td>17</td>
<td>672.4</td>
<td>17.86</td>
<td>55.3%</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>1346.32</td>
<td>42.58</td>
<td>17</td>
<td>716.874</td>
<td>18.89</td>
<td>55.8%</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>45</td>
<td>1344.22</td>
<td>43.17</td>
<td>17</td>
<td>749.8</td>
<td>19.71</td>
<td>54.3%</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>1390.19</td>
<td>44.89</td>
<td>19</td>
<td>741.816</td>
<td>19.75</td>
<td>56%</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>49</td>
<td>1455.69</td>
<td>46.46</td>
<td>21</td>
<td>850.2</td>
<td>22.74</td>
<td>51%</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>50</td>
<td>1515.2</td>
<td>48.48</td>
<td>21</td>
<td>849.4</td>
<td>22.48</td>
<td>53.6%</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>53</td>
<td>1555.25</td>
<td>49.72</td>
<td>23</td>
<td>902.11</td>
<td>23.89</td>
<td>52%</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>55</td>
<td>1608.46</td>
<td>51.4</td>
<td>23</td>
<td>898.75</td>
<td>23.89</td>
<td>54%</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>56</td>
<td>1668.56</td>
<td>53.13</td>
<td>25</td>
<td>876.35</td>
<td>23.32</td>
<td>56.2%</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>58</td>
<td>1765.02</td>
<td>56.48</td>
<td>25</td>
<td>963.95</td>
<td>25.76</td>
<td>54.6%</td>
<td></td>
</tr>
</tbody>
</table>
(a) example 1 - example 5

(b) example 5 - example 8
(c) example 9 - example 12

(d) example 13 - example 16
(e) example 17- example 20

Figure 4.7 Statistic results of drilling simulation in terms of cycle time (20 examples)

The following results can be derived from these simulations:

- The savings of total cycle time is around 50%. Some simulations show a higher saving which is over 55%. There is also a trend: the more the number of holes, the bigger the savings. It is obvious that the Smart Drilling System will display its significant advantage in terms of larger number of holes.

- The savings in time are obtained from the optimal arrangement of tool change and tool travel. Manual drilling performs "haphazardly"; it can only finishes one hole completely and then changes tools to drill the next hole. On the other hand, smart drilling sorts all the holes first and then arranges an optimal drilling sequence to guarantee minimum tool change and travel time.
CHAPTER V
CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, a prototype Smart Drilling System for the process planning of drilling operations was presented. The system has four major functions: 1) drill selection and sequencing 2) cutting condition design 3) NC code generation 4) drilling simulation. Based on the discussion in previous chapters, the following conclusions can be drawn.

In the drill selection and sequencing, the fuzzy set theory is used as a tool to represent the fuzziness associated with all information and input-output relationships. The Fuzzy Decision Table (FDT) selects the drills based on three steps: 1) calculation of fuzzy values by using fuzzy membership function, 2) searching of fuzzy values inside the table, 3) conduction of drill selection.

For the cutting condition design, fuzzy decision making is used to design optimal cutting parameters, cutting speed and feed. A fuzzy operation - intersection is conducted to combine all factors that will affect the cutting condition. Through intersection, the range of optimal cutting conditions can be determined.

The developed system is able to automatically generate NC code in APT format. It can also simulate the drilling operation by using VERICUT®. From the simulation result we see that the Smart Drilling System has obvious advantage: the saving in cycle time is around 50% compared to sequence drilling.
The major reasons for the big saving are the following:

- optimal selection of drill sequence and moving path
- optimal arrangement of tool changes
- optimal design of cutting condition

So far, the Smart Drilling System has been developed and good results have been obtained from simulation examples. There is still some future to be done.

5.2 Future Work

The Smart Drilling System has shown its advantages by the previous simulation results. Yet, in order to be used in the real manufacturing work shop, research is required before the final goal of an entirely automated process planning system can be completed. Future work is listed as follows:

1. The major objective is to reduce machining cost. Hence, there are still many other factors that influence the total cost, such as operator factor, machine factor, etc. Further work should consider all the details of these factors and put them into a cost model.

2. Even though fuzzy logic shows a great advantage, especially when the system faces imprecise information, it still does not work well sometimes. A more robust fuzzy expert system needs to be developed. Work should focus on a better fuzzy algorithm and more efficient fuzzy decision making.
3. In order to adapt the Smart Drilling System to practical utilization, more practical drilling information is required. In addition, developing the learning ability for the system is also necessary.

4. A better user-friendly interface is needed. Implementing this system into a commercial CAD/CAM package would be a good target, which will result in a positive contribution to practical manufacturing.
REFERENCES


Jamsa, K., 1994, *Success with C++*, Jamsa Press, USA.


APPENDIX A: THE SOURCE CODE OF MAIN PROGRAM

/*************************************************************************/
Smart.c  October 18, 1996, this program will automatically generate the process
planning for drilling.

It could perform the following functions:
- open new data file
- retrieve input data of holes
- open existed data file
- read data into the main program
- sort hole data, generate a queue
- generate a drill list
- generate a hole list (including predrill holes)
- match drill list and hole list
- select drills and arrange drill sequence
- generate the process planning for drilling
- generate the APT code for NC machining
- calculate cutting time
*************************************************************************/

#include<stdio.h>
#include<iostream.h>
#include<stdlib.h>
#include<string.h>
#include<alloc.h>
#include<math.h>
#include<fstream.h>
#include<cint.h>

typedef struct hole_rec {
    int hole_num;
    char shape[20];
    float cordi[2];
    float diam;
    float depth;
} hole;

#define SIZE 25

hole part[SIZE];

#define NULL 0
#define LEN sizeof(struct link_list)

#define NUM 26

88
typedef struct drill_condi {
    float diam;
    float speed;
    float feed;
} drcond;

drcond cond[NUM];

// link_list is used to store hole-list(hlist) and drill-list

typedef struct link_list {
    int hole_num;
    char shape[20];
    float cordi[2];
    float diam;
    float depth;
    struct link_list *next;
} list;

list hlist[50], hlistb[50];

// *creat() is a function that generate link_list of holes

struct link_list *creat() /* this function return a pointer to head */{

    list *head,*p1,*p2;
    int n=0;

    p1=p2=(list *)malloc(LEN);
    p1->hole_num = hlist[n].hole_num;
    strcpy(p1->shape , part[n].shape);
    p1->cordi[0]=hlist[n].cordi[0];
    p1->cordi[1]=hlist[n].cordi[1];
    p1->diam =hlist[n].diam;
    p1->depth =hlist[n].depth;

    head = NULL;

    while (p1->hole_num != 0)
    {
        n = n+1;
        if (n == 1) head =p1;
        else p2-> next =p1;
        p2 = p1;
        p1 = (list *)malloc(LEN);
        p1->hole_num = hlist[n].hole_num;

        p1->cordi[0]=hlist[n].cordi[0];
        p1->cordi[1]=hlist[n].cordi[1];
        p1->diam =hlist[n].diam;
        p1->depth =hlist[n].depth;
        p1->hole_num = hlist[n].hole_num;

        p1->shape = part[n].shape;
        p1->shape[0] = part[n].shape[0];
    }

    return head;
}
strcpy(p1->shape, hlist[n].shape);
p1->cordi[0]=hlist[n].cordi[0];
p1->cordi[1]=hlist[n].cordi[1];
p1->diam =hlist[n].diam;
p1->depth =hlist[n].depth;

p2-> next = NULL;
return(head);
}

// the following function *genlist() is used to generate a drill_list
// only consider the direct drills and only diam>2"
// as to the shape drill, there will
// be a independent list about shape drill list

typedef struct drill_list {

    float diam;
    float depth;
    struct drill_list *next;

} drill;

#define LDN sizeof(struct drill_list)

drill dlist[50];
drill dlist1[50];

// *genlist() is a function that generate link_list of drill

drill *genlist() /* this function return a pointer to head */
{
    drill *top,*p1,*p2;
    int n =0;
    p1=p2=(drill *)malloc(LDN);

    p1->diam =dlist[n].diam;
p1->depth =dlist[n].depth;

    top = NULL :

    while (p1->diam != 0)
    {
        n = n+1;
        if (n == 1) top =p1;
        else p2-> next =p1;
p2 = p1;
p1 = (drill *)malloc(LDN);
    p1->diam =dlist[n].diam;
    p1->depth =dlist[n].depth;


p2->next = NULL;
return(top);
}

// cmp( ) function is used for qsort( ) later

int cmp(const void *v1, const void *v2)
{
    hole *h1=(hole*) v1;
    hole *h2=(hole*) v2;

    if(h1->diam > h2->diam)
        return 1;
    else if(h1->diam < h2->diam)
        return -1;
    return 0;
}

// cmpp( ) function is used for qsort( ) hole's shape drill

int i;

int cmpp(const void *v1, const void *v2)
{
    float x1,y1,x2,y2:

    hole *h1=(hole*) v1;
    hole *h2=(hole*) v2;

    x1 = (h1+i)->cord[0];
    y1 = (h1+i)->cord[1];
    x2 = (h2+i)->cord[0];
    y2 = (h2+i)->cord[1];

    if(x1*x1+y1*y2 > x2*x2+y2*y2)
        return 1;
    else if(x1*x1+y1*y2 < x2*x2+y2*y2)
        return -1;
    return 0;
}

// cmp1( ) function is used for qsort( ) in dlist

int cmp1(const void *v1, const void *v2)
{
    drill *h1=(drill*) v1;
    drill *h2=(drill*) v2;
if(h1->diam > h2->diam)
    return 1;
else if(h1->diam < h2->diam)
    return -1;
return 0;
}

// cmp2() function is used for qsort() in hlist

int cmp2(const void *v1, const void *v2)
{
    list *h1=(list*) v1;
    list *h2=(list*) v2;

    if(h1->diam > h2->diam)
        return 1;
    else if(h1->diam < h2->diam)
        return -1;
    return 0;
}

// cmp3() function is used for qsort the nearest hole to the (0,0)
// then set this hole as the first one among the same diam holes

int cmp3(const void *v1, const void *v2)
{
    float x1,y1,x2,y2;

    list *h1=(list*) v1;
    list *h2=(list*) v2;

    x1= (h1+i)->cordi[0];
    y1= (h1+i)->cordi[1];
    x2= (h2+i)->cordi[0];
    y2= (h2+i)->cordi[1];

    if(x1*x1+y1*y1 > x2*x2+y2*y2)
        return 1;
    else if((x1*x1+y1*y12 < x2*x2+y2*y2)
        return -1;
    return 0;
}

// cmp4() is used for the qsort of holes nearest to the first hole

int cmp4(const void *v1, const void *v2)
{
    float x1,y1,x2,y2;

    list *h1=(list*) v1;
    list *h2=(list*) v2;

    if(h1->diam > h2->diam)
        return 1;
    else if(h1->diam < h2->diam)
        return -1;
    return 0;
}
x1= ((h1+i+1)->cordi[0]-(h1+i)->cordi[0]);
y1= ((h1+i+1)->cordi[1]-(h1+i)->cordi[1]);
x2= ((h1+i+2)->cordi[0]-(h1+i)->cordi[0]);
y2= ((h1+i+2)->cordi[1]-(h1+i)->cordi[1]);

if(x1*x1+y1*y2 > x2*x2+y2*y2)
  return 1;
else if(x1*x1+y1*y2 < x2*x2+y2*y2)
  return -1;
return 0;
}

main()
{

int i;

// Open the data file, named "hole", read data into main program

FILE *fp;
fp=fopen("hole","rb");

for(i=0;i<SIZE; i++)
{
  fread(&part[i], sizeof(struct hole_rec),1,fp);
  cout<< part[i].hole_num<< "<<part[i].shape<< "
  cout<<part[i].cordi[0];
  cout<<" "<<part[i].cordi[1]" "<<part[i].diam;
  cout<<" "<<part[i].depth<<"n";
}

// sorting the sequence of holes based on diameters//
// Then generate a new list of holes based on diam (small-> large)
// part[0].hole is the smallest

cout<<"n print sorted holes: 

int j;
int size=SIZE;
qsort(part,size,sizeof(hole),cmp);
for(i=0;i<SIZE;i++)
{
  cout <<"#"<< part[i].hole_num <<"<< part[i].shape<< "
  cout<<"x="<<part[i].cordi[0]<<" in";
  cout<<" y="<< part[i].cordi[1]<<" in";
  cout<<" d=" <<part[i].diam<<" in" << "
  cout <<" depth ="<< part[i].depth<<" in"<<endl<<"n";
}
```cpp
using namespace std;

int main()
{
    // Generate APT code for non-planning drilling
    ofstream bout("testb25.apr", ios::out);

    bout << "VERICUT-apr\n";
    bout << "version-2.0\n";
    bout << "BEGIN\n";
    bout << "CUTTER/.130,....,2\n";
    bout << "SPINDLE/2000\n";
    bout << "SPINDLE/ON,CCL\n";
    bout << "RAPID\n";
    bout << "GOTO/0,0.3\n";
    for (i = 0; i < SIZE; i++)
    {
        bout << "RAPID\n";
        bout << "GOTO/" << part[i].cordi[0] << ",.\n";
        bout << part[i].cordi[1] << ",.<<3\n";
        bout << "FEADRAT/20,IPM\n";
    }
    for (j = 1; j <= i; j++)
    {
        bout << "GOTO/" << part[i].cordi[0] << ",.\n";
        bout << part[i].cordi[1] << ",.<<3-j*0.05<<\n";
    }
    bout << "PAPID\n";
    bout << "GOTO/" << part[i].cordi[0] << ",.\n";
    bout << part[i].cordi[1] << ",.<<3\n";
    bout << "GOTO/0,0.3\n";

    // above finish the predrilling
    // next will consider the direct drilling

    float d1, h1;
    int i;

    for (i = 0; i < SIZE; i++)
    {
        d1 = part[i].diam;
        h1 = part[i].depth;
        if (part[i].diam <= 1)
        {
            bout << "CUTTER/" << d1 << ",...." << h1 << "\n";
            bout << "SPINDL/1000\n";
            bout << "SPINDL/ON,CCL\n";
            bout << "RAPID\n";
            bout << "GOTO/" << part[i].cordi[0] << ",.\n";
            bout << part[i].cordi[1] << ",.<<3.000\n";
            bout << "FEADRAT/50,IPM" << "\n";
        }
    }
    return 0;
}
```
for(r=1;r<41;r++)
{
    bout<<"GOTO/<part[i].cordi[0]>>,",";
    bout<<part[i].cordi[1]<<","<<3.000-r*(part[i].depth)/40<<"\n";
}
}

if(part[i].diam<=2 && part[i].diam>1)
{
    bout<<"CUTTER/"<<.5*d1<<"......"<<h1<<"\n";
    bout<<"SPINDL/1000\n";
    bout<<"SPINDL/ON,CCLW\n";
    bout<<"RAPID\n";
    bout<<"GOTO/<part[i].cordi[0]>>,",";
    bout<<part[i].cordi[1]<<"","<<3.000\n";
    bout<<"FEADRAT/100,IPM"<<"\n";

    for(r=1;r<41;r++)
    {
        bout<<"GOTO/<part[i].cordi[0]>>,",";
        bout<<part[i].cordi[1]<<","<<3.000-r*(part[i].depth)/40<<"\n";
    }

    bout<<"RAPID\n";
    bout<<"GOTO/<part[i].cordi[0]>>,",";
    bout<<part[i].cordi[1]<<"","<<3.000\n";
    bout<<"GOTO/0,0,3"<<"\n";

    bout<<"CUTTER/"<<d1<<"......"<<h1<<"\n";
    bout<<"SPINDL/1000\n";
    bout<<"SPINDL/ON,CCLW\n";
    bout<<"RAPID\n";
    bout<<"GOTO/<part[i].cordi[0]>>,",";
    bout<<part[i].cordi[1]<<"","<<3.000\n";
    bout<<"FEADRAT/100,IPM"<<"\n";

    for(r=1;r<41;r++)
    {
        bout<<"GOTO/<part[i].cordi[0]>>,",";
        bout<<part[i].cordi[1]<<","<<3.000-r*(part[i].depth)/40<<"\n";
    }
}

if(part[i].diam>=2)
{
    bout<<"CUTTER/"<<.5*d1<<"......"<<h1<<"\n";
    bout<<"SPINDL/800\n";
    bout<<"SPINDL/ON,CCLW\n";
    bout<<"RAPID\n";
    bout<<"GOTO/<part[i].cordi[0]>>,",";
    bout<<part[i].cordi[1]<<"","<<3.000\n";

    for(r=1;r<41;r++)
    {
bout<<"FEADRAT/100,IPM"<<"\n";

for(r1=r<41;r++)
{
    bout<<"GOTO/"<<part[i].cordi[0]<<",";
    bout<<part[i].cordi[1]<<","<<3.000-r*(part[i].depth)/40<<"\n";
}

bout<<"RAPID\n";
bout<<"GOTO/"<<part[i].cordi[0]<<",";
bout<<part[i].cordi[1]<<"."<<3.000\n";
bout<<"GOTO/0,0,3"<<"\n";

bout<<"CUTTER/"<<75*d1<<"......,"<<h1<<"\n";
bout<<"SPINDL/800n";
bout<<"SPINDL/ON,CCLW\n";
bout<<"RAPID\n";

bout<<"GOTO/"<<part[i].cordi[0]<<",";
bout<<part[i].cordi[1]<<"."<<3.000\n";
bout<<"FEADRAT/100,IPM"<<"\n";

for(r1=r<41;r++)
{
    bout<<"GOTO/"<<part[i].cordi[0]<<",";
    bout<<part[i].cordi[1]<<"."<<3.000-r*(part[i].depth)/40<<"\n";
}

bout<<"RAPID\n";
bout<<"GOTO/"<<part[i].cordi[0]<<",";
bout<<part[i].cordi[1]<<"."<<3.000\n";
bout<<"GOTO/0,0,3"<<"\n";

bout<<"CUTTER/"<<d1<<"......,"<<h1<<"\n";
bout<<"SPINDL/800n";
bout<<"SPINDL/ON,CCLW\n";
bout<<"RAPID\n";

bout<<"GOTO/"<<part[i].cordi[0]<<",";
bout<<part[i].cordi[1]<<"."<<3.000\n";
bout<<"FEADRAT/100,IPM"<<"\n";

for(r1=r<41;r++)
{
    bout<<"GOTO/"<<part[i].cordi[0]<<",";
    bout<<part[i].cordi[1]<<"."<<3.000-r*(part[i].depth)/40<<"\n";
}

bout<<"RAPID\n";
bout<<"GOTO/"<<part[i].cordi[0]<<",";
bout<<part[i].cordi[1]<<"."<<3.000\n";
bout<<"GOTO/0,0,3"<<"\n";

96
// the following will generate the shape drilling

for(i=0;i<SIZE;i++)
{
    if(strcmp(part[i].shape,"ctb")==0)
    {
        bout<<"CUTTER/"<<1.2*part[i].diam<<"......, \n";
        bout<<"RAPID\n";
        bout<<"GOTO/"<<part[i].cordi[0]<<", ";
        bout<<part[i].cordi[1]<<".3\n";
        bout<<"FEDRAT/20,IPM\n";

        for(r=1;r<11;r++)
        {
            bout<<"GOTO/"<<part[i].cordi[0]<<", ";
            bout<<part[i].cordi[1]<<"<3-r*0.05<<"\n";
        }
    }

    if(strcmp(part[i].shape,"cts")==0)
    {
        bout<<"CUTTER/"<<1.4*part[i].diam<<"30\n","n";
        bout<<"RAPID\n";
        bout<<"GOTO/"<<part[i].cordi[0]<<", ";
        bout<<part[i].cordi[1]<<".3\n";

        bout<<"FEDRAT/30,IPM\n";

        for(r=1;r<11;r++)
        {
            bout<<"GOTO/"<<part[i].cordi[0]<<", ";
            bout<<part[i].cordi[1]<<"<3-r*0.05<"\n";
        }
    }

    bout<<"PAPID\n";
    bout<<"GOTO/"<<part[i].cordi[0]<<", ";
    bout<<part[i].cordi[1]<<"<3\n";
    bout<<"GOTO/0,0,3\n";
}
bout<<"END\n";
bout.close();

/* determine whether pre-drilling is required
  1. holes without pre-drilling are listed and determine the drills
  2. holes with pre-drilling are listed and determine the planning */
/******************************************************************************
/**
  *  the following will generate a list for drills
  */
*******************************************************************************/

// further processing the above data (without predrilling)
// following program will handle the holes with same diameter
// if it is the same diameter, then search the lmax
// once finish search, add one node to the drill-list

int lmax,l;
l=i=0;
cout<<"n drills used for no-predrill holes:\n\n";

while (i<SIZE){
  if (part[i].diam <= 1){
    if(part[i].diam == part[i+4].diam)
    {
      lmax=part[i].depth;
      if (lmax<part[i+1].depth)
        lmax = part[i+1].depth;
      if (lmax < part[i+2].depth)
        lmax = part[i+2].depth;
      if (lmax < part[i+3].depth)
        lmax = part[i+3].depth;
      if (lmax< part[i+4].depth)
        lmax = part[i+4].depth;
      cout<<"5 holes with diam = ";
      cout<<part[i].diam <<" in ";
      cout<< "depth = " << lmax <<" in " <<"\n";
      dlist[l].diam= part[i].diam;
      dlist[l].depth = lmax;
      l = l+1;
      i=i+5;
      continue;
    }
    if (part[i].diam == part[i+3].diam)
    {
      lmax=part[i].depth;
      if (lmax<part[i+1].depth)
        lmax = part[i+1].depth;
      if (lmax < part[i+2].depth)
        lmax = part[i+2].depth;
      if (lmax < part[i+3].depth)
        lmax = part[i+3].depth;
      cout<<"4 holes with diam = ";
      cout<<part[i].diam <<" in ";
      cout<< "depth = " << lmax <<" in " <<"\n";
      dlist[l].diam= part[i].diam;
      dlist[l].depth = lmax;
      l = l+1;
      i=i+4;
    }
  }
  //...
continue;
}
if (part[i].diam == part[i+2].diam)
{
    lmax=part[i].depth;
    if (lmax<part[i+1].depth)
        lmax = part[i+1].depth;
    if (lmax < part[i+2].depth)
        lmax = part[i+2].depth;
    cout<<" 3 holes with diam = ":
        cout<<part[i].diam <<" in ":
        cout<<" depth = " << lmax <<" in " <<"\n";
    dlist[i].diam = part[i].diam;
    dlist[i].depth = lmax;
    l = l+1;
    i=i+3;
    continue;
}
if (part[i].diam == part[i+1].diam)
{
    lmax=part[i].depth;
    if (lmax<part[i+1].depth)
        lmax = part[i+1].depth;
    cout<<" 2 holes with diam = ":
        cout<<part[i].diam <<" in ":
        cout<<" depth = " << lmax <<" in " <<"\n";
    dlist[i].diam = part[i].diam;
    dlist[i].depth = lmax;
    l = l+1;
    i=i+2;
    continue;
} else {
    cout<<" the hole with diam = ":
    cout<< part[i].diam <<" in ":
    cout<<" depth = " << part[i].depth<<" in " <<"\n";
    dlist[i].diam = part[i].diam;
    dlist[i].depth = part[i].depth;
    l = l+1;
}
}
/* generate the fixed drills

1. based on the above work done for holes with diam<=1
2. add drills to the array with diam>1
3. all the drills are fixed in terms of the hole diam required

*******************************************************************************/

for(i=0;i<SIZE;i++){
    if (part[i].diam >1){
        dlist[l].diam = part[i].diam;
        dlist[l].depth = part[i].depth;
        l = l+1;
    }
    else continue;
}

// above work generated the fixed drills

cout<<"\n print dlist using array 1: \n";

i=0;
while(dlist[i].diam != 0){
    cout<<"dlist-array" << " " << dlist[i].diam;
    cout<<" in " << dlist[i].depth<< " in\n";
    i=i+1;
}

cout<<"\n print dlist using array: \n\n";

// check the result of dlist(fixed drills):

i=0;
while(dlist[i].diam != 0){
    cout<<"dlist= " << dlist[i].diam<< " ";
    cout<<dlist[i].depth<< " \n";
    i=i+1;
}

// following deal with drills for holes with 1 predrill
// that is the 1"<diam<=2"

int q=i;
int m=0;
for(j=0;j<SIZE;j++){
    if(part[j].diam>1 && part[j].diam<=2){
        m=m+1;
    }
}

if((part[j].diam/2)>dlist[m].diam && (part[j].diam/2)<=dlist[m+1].diam)
   if((part[j].diam/2 - dlist[m].diam)<0.15)
   {  

if(dlist[m].depth < part[j].depth)
    dlist[m].depth = part[j].depth;

if((dlist[m+1].diam - part[j].diam/2)<0.15)
    { if (dlist[m+1].depth < part[j].depth)
        dlist[m+1].depth = part[j].depth;
    }
else { dlist[q].diam = part[j].diam/2;
        dlist[q].depth = part[j].depth;
        q++;
    }
else continue;
}
else continue;
}

// above work done for 1<diam<2; then check:
// in addition, there is qsort to get the drills in order

qsort(dlist,q,sizeof(drill).cmp1);

i=0;
while(dlist[i].diam !=0){
    cout<<"dlist-chk1" << " " << dlist[i].diam;
    cout<<" in " <<dlist[i].depth<<" in\n";
    i=i+1;
}

cout<<"\n\n";
cout<<"\n";
cout<<"\n\n print the drill list: \n\n";

// print dlist using link_list

drill *top,*t;
top = genlist();
t = top;
if (top !=NULL)
do{
    cout<<" dlist " << t->diam<<" in " << t->depth <<" in \n";
    t = t->next;
}while (t !=NULL);

//generate hlist in which, the predrill holes are examined
//if the hole diam is within the tolerance of .15" then the hole
//could be organized as a fixed drill size.
//Finally, the hlist will have the same diameter with dlist
//by the way, as we already done, dlist decided based on holes
// read data from part[i] into a new type of data: linked list

/* steps: 1. type define new data struct list
   2. read data from part[i] to hole_list[i] */

for (i=0;i<SIZE;i++){
  hlist[i].hole_num = part[i].hole_num;
  strncpy(hlist[i].shape .part[i].shape);
  hlist[i].cord[0] = part[i].cord[0];
  hlist[i].cord[1] = part[i].cord[1];
  hlist[i].diam = part[i].diam;
  hlist[i].depth = part[i].depth;
  cout<< hlist[i].hole_num<< " ";
  cout<< hlist[i].shape<< " ";
  cout<< hlist[i].diam<< " " <<hlist[i].depth<< "\n";
}

struct link list *head,*p;
head = creat();
p=head;
if (head ![= NULL])
do
    cout<< p-> hole_num <<" " << p ->shape<<" " <<p ->cord[0]<<" " ;
    cout<<p -> cord[1] <<" " << p -> diam <<" " <<p->depth<<"\n";
    p = p -> next;
} while(p ![= NULL]);

//above work done for generating a list for all holes

int k=i;
for(j=0;j<SIZE;j++)
{
  if(part[j].diam>1 &&part[j].diam<=2){
    for(m=0;m<SIZE;m++)

    if((part[j].diam/2)>dlist[m].diam && (part[j].diam/2)<=dlist[m+1].diam)
        if((part[j].diam/2 - dlist[m].diam)<0.15)
        {
          hlist[k].hole_num = part[j].hole_num;
          strncpy(hlist[k].shape,"str");
          hlist[k].cord[0]=part[j].cord[0];
          hlist[k].cord[1] = part[j].cord[1];
          hlist[k].diam = dlist[m].diam;
          hlist[k].depth = part[j].depth;
          k=k+1;
  }
if((dlist[m+n].diam - part[j].diam)<0.15)
{
    hlist[k].hole_num = part[j].hole_num;
    strcpy(hlist[k].shape,"str");
    hlist[k].cord[0]=part[j].cord[0];
    hlist[k].cord[1] = part[j].cord[1];
    hlist[k].diam = dlist[m+n].diam;
    hlist[k].depth = part[j].depth;
    k=k+1;
}
else continue;
}
else continue;
}

//deal with predrill holes with diam>2"

int n;
for(i=0;i<SIZE;i++)
{
    if(part[i].diam>2)
    {
        for(n=0;n<SIZE;n++)
        {
            if((part[i].diam/2) > dlist[n].diam & (part[i].diam/2)<dlist[n+1].diam))
                if (((part[i].diam/2 - dlist[n].diam)<0.15)
                {
                    hlist[k].hole_num = part[i].hole_num;
                    strcpy(hlist[k].shape,"str");
                    hlist[k].cord[0]=part[i].cord[0];
                    hlist[k].cord[1] = part[i].cord[1];
                    hlist[k].diam = dlist[n].diam;
                    hlist[k].depth = part[i].depth;
                    k=k+1;
                }
        }
    }
else continue;
}

for(n=0;n<SIZE;n++){
  if((part[i].diam*.75)>dlist[n].diam && (part[i].diam*.75)<=dlist[n+1].diam)
    {
      if((part[i].diam*.75 - dlist[n].diam)<0.15)
        {
          hlist[k].hole_num = part[i].hole_num;
          strcpy(hlist[k].shape,"str");
          hlist[k].cordi[0]=part[i].cordi[0];
          hlist[k].cordi[1] = part[i].cordi[1];
          hlist[k].diam = dlist[n].diam;
          hlist[k].depth = part[i].depth;
          k=k+1;
        }
      if((dlist[n+1].diam - part[i].diam*.75)<0.15)
        {
          hlist[k].hole_num = part[i].hole_num;
          strcpy(hlist[k].shape,"str");
          hlist[k].cordi[0]=part[i].cordi[0];
          hlist[k].cordi[1] = part[i].cordi[1];
          hlist[k].diam = dlist[n+1].diam;
          hlist[k].depth = part[i].depth;
          k=k+1;
        }
      else continue;
    }
  else continue;
}

// check hlist using array

// the following steps is to match the dlist with the hlist
// then the drilling process will be generated based on dlist

// In order to consider the moving sequence, the following issues:
// 1. only multiple holes with the same diam are considered
// 2. among the holes, find the nearest one to point (0,0)
// 3. set this hole as the first in hlist (with d=diam holes)(using cmp3)
// 4. based on this hole, search others to make a right queue
// 5. find the nearest to furthest, make the queue properly(using cmp4)
*/

// first arrange the holes nearby the point(0,0)

cout<<"\n";

for(i=0;i<k;i++)
  {
    cout<<hlist[i].hole_num<<" ";
    cout<<hlist[i].shape<<" "<<hlist[i].cordi[0];
    cout<<" "<<hlist[i].cordi[1]<<" "<<hlist[i].diam;
    cout<<" "<<hlist[i].depth<<" \n";
  
104
// Based on the previous sorted list, find the shape drill sequence

int s,u;
float temp,x1,x2,y1,y2;
int x,y,tem;

// second, make a queue according to the diam

cout<<"\n";
qusort(hlist,k.sizeof(list),cmp2);

for(i=0;i<k;i++)
{
  cout<<chlist[i].hole_num<<" ";
  cout<<chlist[i].shape<<" "<<chlist[i].cordi[0];
  cout<<" "<<chlist[i].cordi[1]<<" "<<chlist[i].diam;
  cout<<" "<<chlist[i].depth<<" \n";
}

// the following will use the new algorithm to deal with the moving
// sequence problem
cout<<"\n\n";

i=0;

while(hlist[i].diam != 0)
{
  j=i;
  u=i;
  if (hlist[j].diam== hlist[j+1].diam)
  {
    x1 = (hlist[j].cordi[0])*(hlist[j].cordi[0]);
    y1 = (hlist[j].cordi[1])*(hlist[j].cordi[1]);
    tem = x1+y1;
    x2 = (hlist[j+1].cordi[0])*(hlist[j+1].cordi[0]);
    y2 = (hlist[j+1].cordi[1])*(hlist[j+1].cordi[1]);
  
    if(tem > (x2+y2))
    {
      tem = x2+y2;

      hlist[50].hole_num = hlist[j+1].hole_num;
      strcpy(hlist[50].shape,hlist[j+1].shape);
      hlist[50].cordi[0]=hlist[j+1].cordi[0];
      hlist[50].cordi[1]=hlist[j+1].cordi[1];
      hlist[50].diam = hlist[j+1].diam;
      hlist[50].depth = hlist[j+1].depth;

      hlist[j+1].hole_num = hlist[i].hole_num;
  
  

strcpy(hlist[j+1].shape,hlist[i].shape);
hlist[j+1].cord[0]=hlist[i].cord[0];
hlist[j+1].cord[1]=hlist[i].cord[1];
hlist[j+1].diam = hlist[i].diam;
hlist[j+1].depth = hlist[i].depth;

hlist[i].hole_num = hlist[50].hole_num;
strcpy(hlist[i].shape,hlist[50].shape);
hlist[i].cord[0]=hlist[50].cord[0];
hlist[i].cord[1]=hlist[50].cord[1];
hlist[i].diam = hlist[50].diam;
hlist[i].depth = hlist[50].depth;
}
j=j+1;

}

if(hlist[i].diam == hlist[i+1].diam)
{
    i=j;
}
else{ i=i+1; }
}

cout<<"\n\n";
for(i=0;i<ck;i++)
{
    x = (hlist[i].cord[0]);
y = (hlist[i].cord[1]);
tem = x*x + y*y;

if(hlist[i].diam == hlist[i+4].diam)
{
    cout<<" the five holes are same";
    cout<<" the diam is " <<hlist[i].diam<<" in \n";

/* following is a bubble sort to make a new queue*/
// try to find the optimal moving way
//find the nearest to (0,0) first, then bubble search

for(j=i;j<i+5;j++)
{
    x1 = (hlist[j].cord[0])*(hlist[j].cord[0]);
y1 = (hlist[j].cord[1])*(hlist[j].cord[1]);
    if(tem >(x1+y1))
    {
        tem = x1+y1;

        hlist[50].hole_num = hlist[j].hole_num;
        strcpy(hlist[50].shape,hlist[j].shape);
    }
}
hlist[50].cordi[0]=hlist[j].cordi[0];
hlist[50].cordi[1]=hlist[j].cordi[1];
hlist[50].diam = hlist[j].diam;
hlist[50].depth = hlist[j].depth;

hlist[j].hole_num = hlist[i].hole_num;
strcpy(hlist[j].shape,hlist[i].shape);
hlist[j].cordi[0]=hlist[i].cordi[0];
hlist[j].cordi[1]=hlist[i].cordi[1];
hlist[j].diam = hlist[i].diam;
hlist[j].depth = hlist[i].depth;

hlist[i].hole_num = hlist[50].hole_num;
strcpy(hlist[i].shape,hlist[50].shape);
hlist[i].cordi[0]=hlist[50].cordi[0];
hlist[i].cordi[1]=hlist[50].cordi[1];
hlist[i].diam = hlist[50].diam;
hlist[i].depth = hlist[50].depth;
}

for(j=i+1;j<i+5;j++)
{
  for(r=i+4;r>=j;r--)
  {
    x1=(hlist[r].cordi[0])-(hlist[j-1].cordi[0]);
y1=(hlist[r].cordi[1])-(hlist[j-1].cordi[1]);
x2=(hlist[r-1].cordi[0])-(hlist[j-1].cordi[0]);
y2=(hlist[r-1].cordi[1])-(hlist[j-1].cordi[1]);
    if((x2*x2+y2*y2)>(x1*x1+y1*y1))
    {
      hlist[50].hole_num = hlist[r-1].hole_num;
      strcpy(hlist[50].shape,hlist[r-1].shape);
      hlist[50].cordi[0]=hlist[r-1].cordi[0];
      hlist[50].cordi[1]=hlist[r-1].cordi[1];
      hlist[50].diam = hlist[r-1].diam;
      hlist[50].depth = hlist[r-1].depth;
      
      hlist[r-1].hole_num = hlist[r].hole_num;
      strcpy(hlist[r-1].shape,hlist[r].shape);
      hlist[r-1].cordi[0]=hlist[r].cordi[0];
      hlist[r-1].cordi[1]=hlist[r].cordi[1];
      hlist[r-1].diam = hlist[r].diam;
      hlist[r-1].depth = hlist[r].depth;
      
      hlist[r].hole_num = hlist[50].hole_num;
      strcpy(hlist[r].shape,hlist[50].shape);
      hlist[r].cordi[0]=hlist[50].cordi[0];
      hlist[r].cordi[1]=hlist[50].cordi[1];
      hlist[r].diam = hlist[50].diam;
    }
hlist[r].depth = hlist[50].depth;
}

if(hlist[i].diam == hlist[i+3].diam)
{
    cout<<" the four holes are same";
    cout<<" the diam is " <<hlist[i].diam<<" in \n";

    for(j=i+1;j<i+4;j++)
    {
        for(r=i+3;r>=j;r--)
        {
            x1=|hlist[r].cordi[0]- (hlist[j-1].cordi[0]);
            y1=|hlist[r].cordi[1]- (hlist[j-1].cordi[1]);
            x2=|hlist[r-1].cordi[0]- (hlist[j-1].cordi[0]);
            y2=|hlist[r-1].cordi[1]- (hlist[j-1].cordi[1]);

            if((x2*x2+y2*y2)>(x1*x1+y1*y1))
            {
                hlist[50].hole_num = hlist[r-1].hole_num;
                strcpy(hlist[50].shape,hlist[r-1].shape);
                hlist[50].cordi[0]=hlist[r-1].cordi[0];
                hlist[50].cordi[1]=hlist[r-1].cordi[1];
                hlist[50].diam = hlist[r-1].diam;
                hlist[50].depth = hlist[r-1].depth;

                hlist[r-1].hole_num = hlist[r].hole_num;
                strcpy(hlist[r-1].shape,hlist[r].shape);
                hlist[r-1].cordi[0]=hlist[r].cordi[0];
                hlist[r-1].cordi[1]=hlist[r].cordi[1];
                hlist[r-1].diam = hlist[r].diam;
                hlist[r-1].depth = hlist[r].depth;

                hlist[r].hole_num = hlist[50].hole_num;
                strcpy(hlist[r].shape,hlist[50].shape);
                hlist[r].cordi[0]=hlist[50].cordi[0];
                hlist[r].cordi[1]=hlist[50].cordi[1];
                hlist[r].diam = hlist[50].diam;
                hlist[r].depth = hlist[50].depth;
            }
        }
    }
}

i=i+4;

108
if(hlist[i].diam == hlist[i+1].diam)
{
    cout<<" the three holes are same";
    cout<<" the diam is " <<hlist[i].diam<<" in \n";
    for(j=i+1;j<i+3;j++)
    {
        for(r=i+2;r>=j;r--)
        {
            x1=(hlist[r].cordi[0]- hlist[j-1].cordi[0]);
            y1=(hlist[r].cordi[1]- hlist[j-1].cordi[1]);
            x2=(hlist[r-1].cordi[0]- hlist[j-1].cordi[0]);
            y2=(hlist[r-1].cordi[1]- hlist[j-1].cordi[1]);

            if((x2*x2+y2*y2)> (x1*x1+y1*y1))
            {

                hlist[50].hole_num = hlist[r-1].hole_num;
                strcpy(hlist[50].shape,hlist[r-1].shape);
                hlist[50].cordi[0]=hlist[r-1].cordi[0];
                hlist[50].cordi[1]=hlist[r-1].cordi[1];
                hlist[50].diam = hlist[r-1].diam;
                hlist[50].depth = hlist[r-1].depth;

                hlist[r-1].hole_num = hlist[r].hole_num;
                strcpy(hlist[r-1].shape,hlist[r].shape);
                hlist[r-1].cordi[0]=hlist[r].cordi[0];
                hlist[r-1].cordi[1]=hlist[r].cordi[1];
                hlist[r-1].diam = hlist[r].diam;
                hlist[r-1].depth = hlist[r].depth;

                hlist[r].hole_num = hlist[50].hole_num;
                strcpy(hlist[r].shape,hlist[50].shape);
                hlist[r].cordi[0]=hlist[50].cordi[0];
                hlist[r].cordi[1]=hlist[50].cordi[1];
                hlist[r].diam = hlist[50].diam;
                hlist[r].depth = hlist[50].depth;
            }
        }        
    }
    i=i+3;
}

if(hlist[i].diam == hlist[i+1].diam)
{
    cout<<" the three holes are same";
    cout<<" the diam is " <<hlist[i].diam<<" in \n";

    for(j=i;j<i+2;j++)
    {
        for(r=i+1;r>=j;r--)
        {
{  
x1=(hlist[r].cordi[0]);  
y1=(hlist[r].cordi[1]);  
x2=(hlist[r-1].cordi[0]);  
y2=(hlist[r-1].cordi[1]);  
  
if((x2*x2+y2*y2)>(x1*x1+y1*y1))  
{
  hlist[50].hole_num = hlist[r-1].hole_num;  
  strcpy(hlist[50].shape,hlist[r-1].shape);  
  hlist[50].cordi[0]=hlist[r-1].cordi[0];  
  hlist[50].cordi[1]=hlist[r-1].cordi[1];  
  hlist[50].diam = hlist[r-1].diam;  
  hlist[50].depth = hlist[r-1].depth;

  hlist[r-1].hole_num = hlist[r].hole_num;  
  strcpy(hlist[r-1].shape,hlist[r].shape);  
  hlist[r-1].cordi[0]=hlist[r].cordi[0];  
  hlist[r-1].cordi[1]=hlist[r].cordi[1];  
  hlist[r-1].diam = hlist[r].diam;  
  hlist[r-1].depth = hlist[r].depth;

  hlist[r].hole_num = hlist[50].hole_num;  
  strcpy(hlist[r].shape,hlist[50].shape);  
  hlist[r].cordi[0]=hlist[50].cordi[0];  
  hlist[r].cordi[1]=hlist[50].cordi[1];  
  hlist[r].diam = hlist[50].diam;  
  hlist[r].depth = hlist[50].depth;
  
}  
}
i=i+2;

}  
}  
else continue;
}  
cout<<"\n\n";

// Open the data file "drill condition", named "drill", read data into main program

cout<<"*************** DRILL CONDITION ***************\n";  
cout<<" DIAM  SPEED  FEED\n";  
FILE *dp;  
dp=fopen("drill","rb");

for(i=0;i<NUM; i++)  
{
  fread (& cond[i], sizeof(struct drill_condi),1,dp);  
cout<<cond[i].diam<<" "<<cond[i].speed;
}
cout="" "<<cond[i].feed<<"n";

}
cout="n";
cout="n
";
cout="n process planning of drilling is the following: \n\n";
t=top;
while(t != NULL){
    for(j=0;j<k;j++)
    {
        if(t->diam == hlist[j].diam){
            cout=" use the drill d="<<t->diam<<" in ";
cout="for hole "<<hlist[j].hole_num<<" "<<hlist[j].shape;
cout=" x="<<hlist[j].cordi[0];
cout=" y="<<hlist[j].cordi[1]<<" dia="<<hlist[j].diam;
cout=" depth="<<hlist[j].depth<<"n";
        }
        else continue;
    }
t=t->next;
}

//up to now, process planning of direct_hole drilling is settled down
qsort(part.size.sizeof(hole).cmp); cout="n\n";
for(i=0;i<SIZE;i++)
{
    cout <<" #< part[i].hole_num for part[i].shape=";
cout=" x="<<part[i].cordi[0]<<" in ";
cout=" y="<<part[i].cordi[1]<<" in ";
cout=" d="<<part[i].diam<<" in "<<" ";
cout=" depth="<<part[i].depth<<" in "<<endl<<"
";
}
cout="n\n";
for(j=1;j<SIZE;j++)
{
    for(r=SIZE-1:r>=j:r--)
    {
        x1=(part[r].cordi[0]- (part[j-1].cordi[0]);
y1=(part[r].cordi[1]- (part[j-1].cordi[1]);
x2=(part[r-1].cordi[0]- (part[j-1].cordi[0]);
y2=(part[r-1].cordi[1]- (part[j-1].cordi[1]);

        if((x2*x2+y2*y2)>(x1*x1+y1*y1))
        {

        


hlist[50].hole_num = part[r-1].hole_num;
strcpy(hlist[50].shape.part[r-1].shape);
part[r-1].shape = hlist[50].shape;
}
}
cout<<"predrill drill sequence\n"

for(i=0;i<SIZE;i++)
{
    cout<<"part[i].hole_num<"<<" ");
    cout<<"part[i].shape<<"<<part[i].cordi[0];
    cout<<" "<<part[i].cordi[1]<<" "<<part[i].diam;
    cout<<" "<<part[i].depth<<" ");
}
cout<<"\n"

//following will generate the APT code used in vericut simulation
ofstream out("test25.apt",ios::out);

out<<"VERICUT-apt\n";
out<<"version-2.0\n";
out<<"BEGIN\n";
out<<"CUTTER/1,30,1,2\n";
out<<"SFINDLE/2000\n";
out<<"SFINDLE/ON,CCLW\n";
out<<"RAPID\n";
for(i=0;i<SIZE;i++)
{
    out<<"RAPID\n";
    out<<"GOTO"<<part[i].cordi[0]<<",";
    out<<part[i].cordi[1]<<""<<"3\n";
out<<"FEDRAT/20.IPM\n";
for (j=1; j<11; j++)
{
    out<<"GOTO/"<<part[i].cord[i][0]<<".";
    out<<part[i].cord[i][1]<<","<<3-j*0.05<<"\n";
}
out<<"RAPID\n";
out<<"GOTO/"<<part[i].cord[i][0]<<".";
out<<part[i].cord[i][1]<<"."<<"3\n";
out<<"GOTO/0.0.3\n";

float d,h;

t = top;
while(t != NULL)
{
    for(r=0; r<NUM;r++)
    {
        if(t->diam == cond[r].diam)
        {
            for(j=0; j<k; j++)
            {
                if(t->diam == hlist[j].diam)
                {
                    d = t->diam;
                    h = t->depth;

                    out<<"CUTTER/"<<d<<"...."<<h<<"\n";
                    out<<"SPINDL/"<<cond[r].speed*2000<<"\n";
                    out<<"SPINDL/ON,CCLW\n";
                    out<<"RAPID\n";

                    out<<"GOTO/"<<chlist[j].cord[i][0]<<".";
                    out<<chlist[j].cord[i][1]<<"."<<"3.000\n";
                    out<<"FEADRAT/"<<cond[r].feed*500<<".IPM"<<"\n";

                    for(s=1; s<41; s++)
                    {
                        out<<"GOTO/"<<chlist[j].cord[i][0]<<".";
                        out<<chlist[j].cord[i][1]<<"."<<3.000-s*(hlist[j].depth)/40<<"\n";
                    }

                    out<<"RAPID\n";
                    out<<"GOTO/"<<chlist[j].cord[i][0]<<".";
                    out<<chlist[j].cord[i][1]<<"."<<"3.000\n";
                }
            }
        }
    }
    while(hlist[j].diam == hlist[j+1].diam)
    {
        out<<"RAPID\n";
        out<<"GOTO/"<<chlist[j+1].cord[i][0]<<".";
    }
}
out<<chlist[j+1].cordi[1]"","<<"3.000\n";
out<<"FEADRAT7"<<condr[1].feed*50<<".IPM"<<"\n";

for(s=1;s<=41;s++)
{
    out<<"GOTO/"<<hlist[j+1].cordi[0]"",";
    out<<hlist[j+1].cordi[1]"",";
    out<<3.000-s*(hlist[j+1].depth)/40<<"\n";
}

out<<"RAPID\n";
out<<"GOTO/"<<hlist[j+1].cordi[0]"",";
out<<hlist[j+1].cordi[1]""<<"3.000\n";

j = j+1;
}
out<<"RAPID\n";
out<<"GOTO/0,0,3\n";
else continue;
}

}

t = t->next;
}

qsort(part.size.sizeof(hole).cmp);
for(i=0;i<SIZE;i++)
{
    cout "#"<< part[i].hole_num <<" "<< part[i].shape <<" ";
    cout<<x = "<< part[i].cordi[0] <<" in";
    cout<<y = "<< part[i].cordi[1] <<" in";
    cout<<d = "<part[i].diam<<" in" ";
    cout<<depth = "<< part[i].depth <<" in"<<endl<<"\n";
}

for(i=0;i<SIZE;i++)
{
    if(strcmp(part[i].shape,"ctb")==0)
    {
        out<<"CUTTER/"<<1.2*part[i].diam<<"",","1\n";
        out<<"RAPID\n";
        out<<"GOTO/"<<part[i].cordi[0]"",";
        out<<part[i].cordi[1]"","3\n";
        out<<"FEDRAT/20,IPM\n";
        for(r=1;r<11;r++)
        {
            out<<"GOTO/"<<part[i].cordi[0]"",";
            out<<part[i].cordi[1]""<<r*0.05<<"\n";
        }
    }
}
} 
out<<"PAPIDn";
out<<"GOTO/"<<part[i].cordi[0]<<",";
out<<"part[i].cordi[1]<<","<<"3\n";

while(part[i].diam == part[i+1].diam)
{
out<<"RAPIDn";
out<<"GOTO/"<<part[i+1].cordi[0]<<",";
out<<"part[i+1].cordi[1]<<","<<"3.000n";
out<<"FEADRAT/20,IPMn";

for(r=1;r<11;r++)
{
out<<"GOTO/"<<part[i+1].cordi[0]<<",";
out<<"part[i+1].cordi[1]<<","<<3-r*0.05<<"n";
}
out<<"RAPIDn";
out<<"GOTO/"<<part[i+1].cordi[0]<<",";
out<<"part[i+1].cordi[1]<<","<<"3.000n";

i=i+1;
}
}

if(strcmp(part[i].shape,"cts")==0)
{
out<<"CUTTER/"<<1.4*part[i].diam<<",30,...,1\n";
out<<"RAPIDn";
out<<"GOTO/"<<part[i].cordi[0]<<",";
out<<"part[i].cordi[1]<<","<<","n";
out<<"FEADRAT/30,IPMn";

for(r=1;r<11;r++)
{
out<<"GOTO/"<<part[i].cordi[0]<<",";
out<<"part[i].cordi[1]<<","<<3-r*0.05<<"n";
}
out<<"PAPIDn";
out<<"GOTO/"<<part[i].cordi[0]<<",";
out<<"part[i].cordi[1]<<","<<"3\n";

while(part[i].diam == part[i+1].diam)
{
out<<"RAPIDn";
out<<"GOTO/"<<part[i+1].cordi[0]<<",";
out<<"part[i+1].cordi[1]<<","<<"3.000n";
out<<"FEADRAT/30,IPMn";

for(r=1;r<11;r++)
115
{  
    out<<"GOTO/"<<part[i+1].cordi[0]<<",";  
    out<<part[i+1].cordi[1]<<","<<3-3*0.05<<"\n";  
}
out<<"RAPID\n";  
out<<"GOTO/"<<part[i+1].cordi[0]<<",";  
out<<part[i+1].cordi[1]<<","<<"3.000\n";  
i=i+1;  
}
out<<"GOTO/0,0,3\n";  
}
out<<"GOTO/0,0,3\n";
}
out<<"END\n";
out.close();

//the following will discuss the cutting time caculation

cout<<"\n ****************************\n";
cout<<"\n list of sorted holes: \n\n";
cout<<"# shape x y diam. depth\n";
qsort(hlist.k.sizeof(list),cmp2);

head = creat();  
p=head;  
if (head != NULL)  
    do  
    {  
        cout<< p-> hole_num <<"  "<< p->shape<<"  "<<p->cordi[0]<<"  ";  
        cout<<p->cordi[1]<<"  "<<p-> diam <<"  "<<p->depth<<"\n";  
        p = p-> next;  
    } while(p != NULL);

double time[50][5],total,a,b;  
double time1[50][5],total1;

// the following will consider the overlap elements

cout<<"\n ****************************\n";
cout<<"\n\n list of cutting time with planning\n\n";
cout<< "step Tmov Tdep Tsha Tcha Total(sec.)\n";

j=0,k=1;  
p=head;  
while(p !=NULL){  
    if(hlist[j].diam == hlist[j+4].diam)  
    {  
        time[j][0]=sqrt((p->cordi[0])* (p->cordi[0]) + (p->cordi[1])* (p->cordi[1]))/0.5;  
        time[j][1]=hlist[i].depth/0.1;

116
if (strcmp(p->shape,"str"))
{
    time[j][2] = 10;
    time[j][3] = 80;
}
else
{
    time[j][2] = 0;
    time[j][3] = 40;
}

cout<<"#"<<k<<" "<<time[j][0]<<" "<<time[j][1];
cout<<" "<<time[j][2]<<" ";
cout<<time[j][3]<<" "<<time[j][4]<<"\n";
p=p->next;

for(i=j+1;i<=j+5;i++){
    a= (hlist[i].cord[0]-hlist[i-1].cord[0]);
    b= (hlist[i].cord[1]-hlist[i-1].cord[1]);
    time[i][0] = sqrt(a*a+b*b)/0.5;
    time[i][1]=hlist[i].depth/0.1;
    if (strcmp(p->shape,"str"))
    {
        time[i][2] = 10;
        time[i][3] = 0;
    }
    else
    {
        time[i][2] = 0;
        time[i][3] = 0;
    }
time[i][4] = time[i][0]+time[i][1]+time[i][2]+time[i][3]+time[i][4];

cout<<"#"<<k<<" "<<time[i][0]<<" "<<time[i][1];
cout<<" "<<time[i][2]<<" ";
cout<<time[i][3]<<" "<<time[i][4]<<"\n";
p=p->next;
}
j=j+5;
k=k+1;
}

if(hlist[j].diam == hlist[j+3].diam)
{
    time[j][0]=sqrt((p->cord[0])*(p->cord[0]) + (p->cord[1])*(p->cord[1]))/0.5;
    time[j][1]=hlist[i].depth/0.1;
    if (strcmp(p->shape,"str"))
    {
        time[j][2] = 10;
        time[j][3] = 80;
    }
    else
```c++
{ 
    time[j][2] = 0;
    time[j][3] = 40;
}


cout<< " #"<<k<<" "<<time[j][0]<<" "<<time[j][1];
cout<<" "<<time[j][2]<<" ":
cout<<time[j][3]<<" "<<time[j][4]<<"\n";
p=p->next;
for(i=i+1;i<j+4;i++)
{a= (hlist[i].cordi[0]-hlist[i-1].cordi[0]);
b= (hlist[i].cordi[1]-hlist[i-1].cordi[1]);
time[i][0] = sqrt(a*a+b*b)/0.5;

time[i][1]=hlist[i].depth/0.1;
if (strcmp(p->shape,"str"))
{ 
    time[i][2] = 10;
    time[i][3] = 0;
}
else
{ 
    time[i][2] = 0;
    time[i][3] = 0;
}
    time[i][4] = time[i][0]+time[i][1]+time[i][2]+time[i][3]+time[i][4];

cout<< " #"<<k<<" "<<time[i][0]<<" "<<time[i][1];
cout<<" "<<time[i][2]<<" ":
cout<<time[i][3]<<" "<<time[i][4]<<"\n";
p=p->next;
}
j=j+4;
k=k+1;
}

if(hlist[j].diam == hlist[j+1].diam)
{ 
    time[j][0]=sqrt((p->cordi[0])*(p->cordi[0]) + (p->cordi[1])*(p->cordi[1]))/0.5;
    time[j][1]=hlist[i].depth/0.1;
if (strcmp(p->shape,"str"))
{ 
    time[j][2] = 10;
    time[j][3] = 80;
}
else
{ 
    time[j][2] = 0;
    time[j][3] = 40;
}
cout<< " #"<<k<<" "<<time[j][0]<<" "<<time[j][1];
```

118
cout<<" <<time[j][2]<< " : 

cout<<time[j][3]<< " <<time[j][4]<<"n"; 
p=p->next; 
for(i=j+1;i<j+2;i++) 
a= (hlist[i].cord[0]-hlist[i-1].cord[0]); 
b= (hlist[i].cord[1]-hlist[i-1].cord[1]); 
time[i][0] = sqrt(a*a+b*b)/0.5; 

time[i][1]=hlist[i].depth/0.1; 
if (strcmp(p->shape."str")) 
{ 
time[i][2] = 10; 
time[i][3] = 0; 
} 
else 
{ 
time[i][2] = 0; 
time[i][3] = 0; 
} 
time[i][4] = time[i][0]+time[i][1]+time[i][2]+time[i][3]+time[i][4]; 
cout<< " #<<k<<" <<time[i][0]<< " <<time[i][1]; 
cout<<" <<time[i][2]<< " ; 
cout<<time[i][3]<< " <<time[i][4]<<"n"; 
p=p->next; 
}
j=j+2; 
k=k+1; 
}

else{ 
time[j][0]=sqrt((p->cord[0])*(p->cord[0]) + (p->cord[1])*(p->cord[1]))/0.5; 
time[j][1]=hlist[j].depth/0.1; 
if (strcmp(p->shape."str")) 
{ 
time[j][2] = 10; 
time[j][3] = 80; 
} 
else 
{ 
time[j][2] = 0; 
time[j][3] = 40; 
} 
cout<< " #<<k<<" <<time[j][0]<< " <<time[j][1]<< " ; 
cout<<time[j][2]<< " <<time[j][3]<< " <<time[j][4]<<"n"; 
j=j+1; 
k=k+1; 
p=p->next; 
} 
}
cout<<"\n\n";
total=0;
for(i=0;i<j;i++)
{
total=total+time[i][4];
}
cout<<" total cutting time is:\n";
cout<<total<<" seconds\n";

cout<<"\n\n******************************\n";
cout<<"list of cutting time without planning:\n\n";
cout<< "step Tmov Tdep Tsha Tcha Total(second):\n";

// if there is no consideration of planning
j=0;
p=head;
while(p !=NULL)
{
time1[j][0]=sqrt((p->cordi[0])*(p->cordi[0]) + (p->cordi[1])*(p->cordi[1]))/0.5;
time1[j][1]=hlist[j].depth/0.1;
if (strcmp(p->shape,"str"))
{
time1[j][2] = 10;
time1[j][3] = 80;
}
else
{
time1[j][2] = 0;
time1[j][3] = 40;
}
cout<<"#"<<j+1"" "<< time1[j][0]"" "<<time1[j][1]<<" "<<time1[j][2];
cout<<"" "<<time1[j][3]"" "<<time1[j][4]<<"\n";
j=j+1;
p=p->next;
}
cout<<"\n\n";
total=0;
for(i=0;i<j;i++)
{
total=total+time[i][4];
}
cout<<" total cutting time(no planning) is:\n";
cout<<total<<" seconds\n";

return 0;
}
VERICUT-apt
version-2.0
BEGIN
CUTTER/.1,.30,....2
SPINDLE/2000
SPINDLE/ON.CCLW
RAPID
GOTO/-1.1,-.1.3
RAPID
GOTO/1,1.3
FEDRAT/20,IPM
GOTO/1,1.2.95
GOTO/-1,1.2.9
GOTO/1,1.2.85
GOTO/1,1.2.8
GOTO/1,1.2.75
GOTO/1,1.2.7
GOTO/1,1.2.65
GOTO/1,1.2.6
GOTO/1,1.2.55
GOTO/1,1.2.5
PAPID
GOTO/1,1.3
RAPID
GOTO/2,2,3
FEDRAT/20,IPM
GOTO/2,2,2.95
GOTO/2,2,2.9
GOTO/2,2,2.85
GOTO/2,2,2.8
GOTO/2,2,2.75
GOTO/2,2,2.7
GOTO/2,2,2.65
GOTO/2,2,2.6
GOTO/2,2,2.55
GOTO/2,2,2.5
PAPID
GOTO/2,2,3
RAPID
GOTO/2,3,3
FEDRAT/20,IPM
GOTO/2,3,2.95
GOTO/2,3,2.9
GOTO/2,3,2.85
GOTO/2,3,2.8
GOTO/2,3,2.75
GOTO/2,3,2.7
GOTO/2,3,2.65
GOTO/2,3,2.6
GOTO/2,3,2.55
GOTO/2,3,2.5
PAPID
GOTO/2,3,3
RAPID
GOTO/4,4,3
FEDRAT/20,IPM
GOTO/4,4,2.95
GOTO/4,4,2.9
GOTO/4,4,2.85
GOTO/4,4,2.8
GOTO/4,4,2.75
GOTO/4,4,2.7
GOTO/4,4,2.65
GOTO/4,4,2.6
GOTO/4,4,2.55
GOTO/4,4,2.5
PAPID
GOTO/4,4,3
RAPID
GOTO/5,2,3
FEDRAT/20,IPM
GOTO/5,2,2.95
GOTO/5,2,2.9
GOTO/5,2,2.85
GOTO/5,2,2.8
GOTO/5,2,2.75
GOTO/5,2,2.7
GOTO/5,2,2.65
GOTO/5,2,2.6
GOTO/5,2,2.55
GOTO/5,2,2.5
PAPID
GOTO/5,2,3
RAPID
GOTO/7,2,3
FEDRAT/20,IPM
GOTO/7,2,2.95
GOTO/7,2,2.9
GOTO/7,2,2.85
GOTO/7,2,2.8
GOTO/7,2,2.75
GOTO/7,2,2.7
GOTO/7,2,2.65
GOTO/7,2,2.6
GOTO/7,2,2.55
GOTO/7,2,2.5
PAPID
GOTO/7,2,3
RAPID
GOTO/9,2,3
FEDRAT/20,IPM
GOTO/9,2,2.95
GOTO/9,2,2.9
GOTO/9,2,2.85
GOTO/9,2,2.8
GOTO/9,2,2.75
GOTO/9,2,2.7
GOTO/9,2,2.65
GOTO/9,2,2.6
GOTO/9,2,2.55
GOTO/9,2,2.5
PAPID
GOTO/9,2,3
RAPID
GOTO/9,3,3
FEDRAT/20,IPM
GOTO/9,3,2.95
GOTO/9,3,2.9
GOTO/9,3,2.85
GOTO/9,3,2.8
GOTO/9,3,2.75
GOTO/9,3,2.7
GOTO/9,3,2.65
GOTO/9,3,2.6
GOTO/9,3,2.55
GOTO/9,3,2.5
PAPID
GOTO/9,3,3
RAPID
GOTO/10,2,3
FEDRAT/20,IPM
GOTO/10,2,2.95
GOTO/10,2,2.9
GOTO/10,2,2.85
GOTO/10,2,2.8
GOTO/10,2,2.75
GOTO/10,2,2.7
GOTO/10,2,2.65
GOTO/10,2,2.6
GOTO/10,2,2.55
GOTO/10,2,2.5
PAPID
GOTO/10,2,3
RAPID
GOTO/9,5,3
FEDRAT/20,IPM
GOTO/9,5,2.95
GOTO/9,5,2.9
GOTO/9,5,2.85
GOTO/9,5,2.8
GOTO/9,5,2.75
GOTO/9,5,2.7
GOTO/9,5,2.65
GOTO/9,5,2.6
GOTO/9,5,2.55
GOTO/9,5,2.5

123
GOTO/2,7,3
RAPID
GOTO/2,9,3
FEDRAT/20,IPM
GOTO/2,9,2,95
GOTO/2,9,2,9
GOTO/2,9,2,85
GOTO/2,9,2,8
GOTO/2,9,2,75
GOTO/2,9,2,7
GOTO/2,9,2,65
GOTO/2,9,2,6
GOTO/2,9,2,55
GOTO/2,9,2,5
PAPID
GOTO/2,9,3
RAPID
GOTO/1,10,3
FEDRAT/20,IPM
GOTO/1,10,2,95
GOTO/1,10,2,9
GOTO/1,10,2,85
GOTO/1,10,2,8
GOTO/1,10,2,75
GOTO/1,10,2,7
GOTO/1,10,2,65
GOTO/1,10,2,6
GOTO/1,10,2,55
GOTO/1,10,2,5
PAPID
GOTO/1,10,3
RAPID
GOTO/5,6,3
FEDRAT/20,IPM
GOTO/5,6,2,95
GOTO/5,6,2,9
GOTO/5,6,2,85
GOTO/5,6,2,8
GOTO/5,6,2,75
GOTO/5,6,2,7
GOTO/5,6,2,65
GOTO/5,6,2,6
GOTO/5,6,2,55
GOTO/5,6,2,5
PAPID
GOTO/5,6,3
GOTO/-,..-05,3
RAPID
GOTO/-0.125,-0.125,0
CUTTER/0.25,.,.,.1
SPINDL/1000
SPINDL/ON,CCLW
RAPID
GOTO/9.2.2.04
GOTO/9.2.2.01
GOTO/9.2.1.98
GOTO/9.2.1.95
GOTO/9.2.1.92
GOTO/9.2.1.89
GOTO/9.2.1.86
GOTO/9.2.1.83
GOTO/9.2.1.8
RAPID
GOTO/9.2.3.000
RAPID
GOTO/10.2.3.000
FEADRAT/50.IPM
GOTO/10.2.2.96
GOTO/10.2.2.92
GOTO/10.2.2.88
GOTO/10.2.2.84
GOTO/10.2.2.8
GOTO/10.2.2.76
GOTO/10.2.2.72
GOTO/10.2.2.68
GOTO/10.2.2.64
GOTO/10.2.2.6
GOTO/10.2.2.56
GOTO/10.2.2.52
GOTO/10.2.2.48
GOTO/10.2.2.44
GOTO/10.2.2.4
GOTO/10.2.2.36
GOTO/10.2.2.32
GOTO/10.2.2.28
GOTO/10.2.2.24
GOTO/10.2.2.2
GOTO/10.2.2.16
GOTO/10.2.2.12
GOTO/10.2.2.08
GOTO/10.2.2.04
GOTO/10.2.2
GOTO/10.2.1.96
GOTO/10.2.1.92
GOTO/10.2.1.88
GOTO/10.2.1.84
GOTO/10.2.1.8
GOTO/10.2.1.76
GOTO/10.2.1.72
GOTO/10.2.1.68
GOTO/10.2.1.64
GOTO/10.2.1.6
GOTO/10.2.1.56
GOTO/10.2.1.52
GOTO/10.2.1.48
GOTO/10.2.1.44

132
GOTO/10,2,1.4
RAPID
GOTO/10,2,3,000
RAPID
GOTO/-0.175,-0.175,3.000
RAPID
GOTO/-0.25,-0.25,0
CUTTER/0.5,....,2
SPINDL/1000
SPINDL/ON,CCLW
RAPID
GOTO/-0.25,-0.25,3
GOTO/2,3,3,000
FEADRAT/50,IPM
GOTO/2,3,2,975
GOTO/2,3,2,95
GOTO/2,3,2,925
GOTO/2,3,2,9
GOTO/2,3,2,875
GOTO/2,3,2,85
GOTO/2,3,2,825
GOTO/2,3,2,8
GOTO/2,3,2,775
GOTO/2,3,2,75
GOTO/2,3,2,725
GOTO/2,3,2,7
GOTO/2,3,2,675
GOTO/2,3,2,65
GOTO/2,3,2,625
GOTO/2,3,2,6
GOTO/2,3,2,575
GOTO/2,3,2,55
GOTO/2,3,2,525
GOTO/2,3,2,5
GOTO/2,3,2,475
GOTO/2,3,2,45
GOTO/2,3,2,425
GOTO/2,3,2,4
GOTO/2,3,2,375
GOTO/2,3,2,35
GOTO/2,3,2,325
GOTO/2,3,2,3
GOTO/2,3,2,275
GOTO/2,3,2,25
GOTO/2,3,2,225
GOTO/2,3,2,2
GOTO/2,3,2,175
GOTO/2,3,2,15
GOTO/2,3,2,125
GOTO/2,3,2,1
GOTO/2,3,2,075
GOTO/2,3,2,05
GOTO/2,3,2,025
GOTO/2.3.2
RAPID
GOTO/2.3.3.000
RAPID
GOTO/2.7.3.000
FEADRAT/50.1PM
GOTO/2.7.2.945
GOTO/2.7.2.89
GOTO/2.7.2.835
GOTO/2.7.2.78
GOTO/2.7.2.725
GOTO/2.7.2.67
GOTO/2.7.2.615
GOTO/2.7.2.56
GOTO/2.7.2.505
GOTO/2.7.2.45
GOTO/2.7.2.395
GOTO/2.7.2.34
GOTO/2.7.2.285
GOTO/2.7.2.23
GOTO/2.7.2.175
GOTO/2.7.2.12
GOTO/2.7.2.065
GOTO/2.7.2.01
GOTO/2.7.1.955
GOTO/2.7.1.9
GOTO/2.7.1.845
GOTO/2.7.1.79
GOTO/2.7.1.735
GOTO/2.7.1.68
GOTO/2.7.1.625
GOTO/2.7.1.57
GOTO/2.7.1.515
GOTO/2.7.1.46
GOTO/2.7.1.405
GOTO/2.7.1.35
GOTO/2.7.1.295
GOTO/2.7.1.24
GOTO/2.7.1.185
GOTO/2.7.1.13
GOTO/2.7.1.075
GOTO/2.7.1.02
GOTO/2.7.0.965
GOTO/2.7.0.91
GOTO/2.7.0.855
GOTO/2.7.0.8
RAPID
GOTO/2.7.3.000
RAPID
GOTO/3.8.3.000
FEADRAT/50.1PM
GOTO/3.8.2.955
GOTO/3.8.2.91
GOTO/7.9.2.67
GOTO/7.9.2.64
GOTO/7.9.2.61
GOTO/7.9.2.58
GOTO/7.9.2.55
GOTO/7.9.2.52
GOTO/7.9.2.49
GOTO/7.9.2.46
GOTO/7.9.2.43
GOTO/7.9.2.4
GOTO/7.9.2.37
GOTO/7.9.2.34
GOTO/7.9.2.31
GOTO/7.9.2.28
GOTO/7.9.2.25
GOTO/7.9.2.22
GOTO/7.9.2.19
GOTO/7.9.2.16
GOTO/7.9.2.13
GOTO/7.9.2.1
GOTO/7.9.2.07
GOTO/7.9.2.04
GOTO/7.9.2.01
GOTO/7.9.1.98
GOTO/7.9.1.95
GOTO/7.9.1.92
GOTO/7.9.1.89
GOTO/7.9.1.86
GOTO/7.9.1.83
GOTO/7.9.1.8
RAPID
GOTO/7.9.3.000
RAPID
GOTO/7.2.3.000
FEADRAD/50,IPM
GOTO/7.2.2.96
GOTO/7.2.2.92
GOTO/7.2.2.88
GOTO/7.2.2.84
GOTO/7.2.2.8
GOTO/7.2.2.76
GOTO/7.2.2.72
GOTO/7.2.2.68
GOTO/7.2.2.64
GOTO/7.2.2.6
GOTO/7.2.2.56
GOTO/7.2.2.52
GOTO/7.2.2.48
GOTO/7.2.2.44
GOTO/7.2.2.4
GOTO/7.2.2.36
GOTO/7.2.2.32
GOTO/7.2.2.28

136
GOTO/7.2.2.24
GOTO/7.2.2.2
GOTO/7.2.2.16
GOTO/7.2.2.12
GOTO/7.2.2.08
GOTO/7.2.2.04
GOTO/7.2.2
GOTO/7.2.1.96
GOTO/7.2.1.92
GOTO/7.2.1.88
GOTO/7.2.1.84
GOTO/7.2.1.8
GOTO/7.2.1.76
GOTO/7.2.1.72
GOTO/7.2.1.68
GOTO/7.2.1.64
GOTO/7.2.1.6
GOTO/7.2.1.56
GOTO/7.2.1.52
GOTO/7.2.1.48
GOTO/7.2.1.44
GOTO/7.2.1.4
RAPID
GOTO/7.2.3.000
RAPID
GOTO/-0.25,-0.25,3.000
RAPID
GOTO/-0.375,-0.375,0
CUTTER/0.75,.....,3
SPINDL/1000
SPINDL/ON,CCLW
RAPID
GOTO/-0.375,-0.375,3
GOTO/5.2.3.000
FEADRA/750,IPM
GOTO/5.2.2.925
GOTO/5.2.2.85
GOTO/5.2.2.775
GOTO/5.2.2.7
GOTO/5.2.2.625
GOTO/5.2.2.55
GOTO/5.2.2.475
GOTO/5.2.2.4
GOTO/5.2.2.325
GOTO/5.2.2.25
GOTO/5.2.2.175
GOTO/5.2.2.1
GOTO/5.2.2.025
GOTO/5.2.1.95
GOTO/5.2.1.875
GOTO/5.2.1.8
GOTO/5.2.1.725
GOTO/5.2.1.65
GOTO/5.2,1.575
GOTO/5.2,1.5
GOTO/5.2,1.425
GOTO/5.2,1.35
GOTO/5.2,1.275
GOTO/5.2,1.2
GOTO/5.2,1.125
GOTO/5.2,1.05
GOTO/5.2,0.975
GOTO/5.2,0.9
GOTO/5.2,0.825
GOTO/5.2,0.75
GOTO/5.2,0.675
GOTO/5.2,0.6
GOTO/5.2,0.525
GOTO/5.2,0.45
GOTO/5.2,0.375
GOTO/5.2,0.3
GOTO/5.2,0.225
GOTO/5.2,0.15
GOTO/5.2,0.075
GOTO/5.2,0
RAPID
GOTO/5.2,3.000
RAPID
GOTO/8,8,3.000
FEADGAT/50,IPM
GOTO/8,8,2.925
GOTO/8,8,2.85
GOTO/8,8,2.775
GOTO/8,8,2.7
GOTO/8,8,2.625
GOTO/8,8,2.55
GOTO/8,8,2.475
GOTO/8,8,2.4
GOTO/8,8,2.325
GOTO/8,8,2.25
GOTO/8,8,2.175
GOTO/8,8,2.1
GOTO/8,8,2.025
GOTO/8,8,1.95
GOTO/8,8,1.875
GOTO/8,8,1.8
GOTO/8,8,1.725
GOTO/8,8,1.65
GOTO/8,8,1.575
GOTO/8,8,1.5
GOTO/8,8,1.425
GOTO/8,8,1.35
GOTO/8,8,1.275
GOTO/8,8,1.2
GOTO/8,8,1.125
GOTO/8,8,1.05

138
GOTO/8.8,0.975
GOTO/8.8,0.9
GOTO/8.8,0.825
GOTO/8.8,0.75
GOTO/8.8,0.675
GOTO/8.8,0.6
GOTO/8.8,0.525
GOTO/8.8,0.45
GOTO/8.8,0.375
GOTO/8.8,0.3
GOTO/8.8,0.225
GOTO/8.8,0.15
GOTO/8.8,0.075
GOTO/8.8,0
RAPID
GOTO/8.8,3.000
RAPID
GOTO/11,10,3.000
FEADRAT/50,1PM
GOTO/11,10,2.9375
GOTO/11,10,2.875
GOTO/11,10,2.8125
GOTO/11,10,2.75
GOTO/11,10,2.6875
GOTO/11,10,2.625
GOTO/11,10,2.5625
GOTO/11,10,2.5
GOTO/11,10,2.4375
GOTO/11,10,2.375
GOTO/11,10,2.3125
GOTO/11,10,2.25
GOTO/11,10,2.1875
GOTO/11,10,2.125
GOTO/11,10,2.0625
GOTO/11,10,2
GOTO/11,10,1.9375
GOTO/11,10,1.875
GOTO/11,10,1.8125
GOTO/11,10,1.75
GOTO/11,10,1.6875
GOTO/11,10,1.625
GOTO/11,10,1.5625
GOTO/11,10,1.5
GOTO/11,10,1.4375
GOTO/11,10,1.375
GOTO/11,10,1.3125
GOTO/11,10,1.25
GOTO/11,10,1.1875
GOTO/11,10,1.125
GOTO/11,10,1.0625
GOTO/11,10,1
GOTO/11,10,0.9375
GOTO/11,10,0.875
GOTO/11.10.0.8125
GOTO/11.10.0.75
GOTO/11.10.0.6875
GOTO/11.10.0.625
GOTO/11.10.0.5625
GOTO/11.10.0.5
RAPID
GOTO/11.10.3.000
RAPID
GOTO/4.9.3.000
FEADRA7/50,1PM
GOTO/4.9.2.925
GOTO/4.9.2.85
GOTO/4.9.2.775
GOTO/4.9.2.7
GOTO/4.9.2.625
GOTO/4.9.2.55
GOTO/4.9.2.475
GOTO/4.9.2.4
GOTO/4.9.2.325
GOTO/4.9.2.25
GOTO/4.9.2.175
GOTO/4.9.2.1
GOTO/4.9.2.025
GOTO/4.9.1.95
GOTO/4.9.1.875
GOTO/4.9.1.8
GOTO/4.9.1.725
GOTO/4.9.1.65
GOTO/4.9.1.575
GOTO/4.9.1.5
GOTO/4.9.1.425
GOTO/4.9.1.35
GOTO/4.9.1.275
GOTO/4.9.1.2
GOTO/4.9.1.125
GOTO/4.9.1.05
GOTO/4.9.0.975
GOTO/4.9.0.9
GOTO/4.9.0.825
GOTO/4.9.0.75
GOTO/4.9.0.675
GOTO/4.9.0.6
GOTO/4.9.0.525
GOTO/4.9.0.45
GOTO/4.9.0.375
GOTO/4.9.0.3
GOTO/4.9.0.225
GOTO/4.9.0.15
GOTO/4.9.0.075
GOTO/4.9.0
RAPID
GOTO/4.9.3.000
RAPID
GOTO/-0.375,-0.375,3.000
RAPID
GOTO/-0.5,-0.5,0
CUTTER/1,....,3
SPINDL/1000
SPINDL/ON,CCLW
RAPID
GOTO/-0.5,-0.5,3
GOTO/5,2,3.000
FEADRAT/100,IPM
GOTO/5,2,2.925
GOTO/5,2,2.85
GOTO/5,2,2.775
GOTO/5,2,2.7
GOTO/5,2,2.625
GOTO/5,2,2.55
GOTO/5,2,2.5
GOTO/5,2,2.475
GOTO/5,2,2.4
GOTO/5,2,2.325
GOTO/5,2,2.25
GOTO/5,2,2.175
GOTO/5,2,2.1
GOTO/5,2,2.025
GOTO/5,2,1.95
GOTO/5,2,1.875
GOTO/5,2,1.8
GOTO/5,2,1.725
GOTO/5,2,1.65
GOTO/5,2,1.575
GOTO/5,2,1.5
GOTO/5,2,1.425
GOTO/5,2,1.35
GOTO/5,2,1.35
GOTO/5,2,1.275
GOTO/5,2,1.2
GOTO/5,2,1.125
GOTO/5,2,1.05
GOTO/5,2,0.975
GOTO/5,2,0.9
GOTO/5,2,0.825
GOTO/5,2,0.75
GOTO/5,2,0.675
GOTO/5,2,0.6
GOTO/5,2,0.525
GOTO/5,2,0.45
GOTO/5,2,0.375
GOTO/5,2,0.3
GOTO/5,2,0.225
GOTO/5,2,0.15
GOTO/5,2,0.075
GOTO/5,2,0
RAPID
GOTO/5,2,3.000
RAPID
GOTO/5,6,3.000
FEADRAT/100,IPM
GOTO/5,6,2.925
GOTO/5,6,2.85
GOTO/5,6,2.775
GOTO/5,6,2.7
GOTO/5,6,2.625
GOTO/5,6,2.55
GOTO/5,6,2.475
GOTO/5,6,2.4
GOTO/5,6,2.325
GOTO/5,6,2.25
GOTO/5,6,2.175
GOTO/5,6,2.1
GOTO/5,6,2.025
GOTO/5,6,1.95
GOTO/5,6,1.875
GOTO/5,6,1.8
GOTO/5,6,1.725
GOTO/5,6,1.65
GOTO/5,6,1.575
GOTO/5,6,1.5
GOTO/5,6,1.425
GOTO/5,6,1.35
GOTO/5,6,1.275
GOTO/5,6,1.2
GOTO/5,6,1.125
GOTO/5,6,1.05
GOTO/5,6,0.975
GOTO/5,6,0.9
GOTO/5,6,0.825
GOTO/5,6,0.75
GOTO/5,6,0.675
GOTO/5,6,0.6
GOTO/5,6,0.525
GOTO/5,6,0.45
GOTO/5,6,0.375
GOTO/5,6,0.3
GOTO/5,6,0.225
GOTO/5,6,0.15
GOTO/5,6,0.075
GOTO/5,6,0
RAPID
GOTO/5,6,3.000
RAPID
GOTO/2,9,3.000
FEADRAT/100,IPM
GOTO/2,9,2.9375
GOTO/2,9,2.875
GOTO/2,9,2.8125
GOTO/2,9,2.75
GOTO/2,9,2.6875
GOTO/8,8,1.35
GOTO/8,8,1.275
GOTO/8,8,1.2
GOTO/8,8,1.125
GOTO/8,8,1.05
GOTO/8,8,0.975
GOTO/8,8,0.9
GOTO/8,8,0.825
GOTO/8,8,0.75
GOTO/8,8,0.675
GOTO/8,8,0.6
GOTO/8,8,0.525
GOTO/8,8,0.45
GOTO/8,8,0.375
GOTO/8,8,0.3
GOTO/8,8,0.225
GOTO/8,8,0.15
GOTO/8,8,0.075
GOTO/8,8,0
RAPID
GOTO/8,8,3.000
RAPID
GOTO/-0.75,-0.75.3.000
RAPID
GOTO/-0.875,-0.875.0
CUTTER/1.75......3
SPINDL/800
SPINDL/ON,CCLW
RAPID
GOTO/-0.875,-0.875.3
GOTO/5,2,3.000
FEADRAT/100,IPM
GOTO/5,2,2.925
GOTO/5,2,2.85
GOTO/5,2,2.775
GOTO/5,2,2.7
GOTO/5,2,2.625
GOTO/5,2,2.55
GOTO/5,2,2.475
GOTO/5,2,2.4
GOTO/5,2,2.325
GOTO/5,2,2.25
GOTO/5,2,2.175
GOTO/5,2,2.1
GOTO/5,2,2.025
GOTO/5,2,1.95
GOTO/5,2,1.875
GOTO/5,2,1.8
GOTO/5,2,1.725
GOTO/5,2,1.65
GOTO/5,2,1.575
GOTO/5,2,1.5
GOTO/5,2,1.425
GOTO/5.6.0.75
GOTO/5.6.0.675
GOTO/5.6.0.6
GOTO/5.6.0.525
GOTO/5.6.0.45
GOTO/5.6.0.375
GOTO/5.6.0.3
GOTO/5.6.0.225
GOTO/5.6.0.15
GOTO/5.6.0.075
GOTO/5.6.0
RAPID
GOTO/5.6.3.000
RAPID
GOTO/-0.875,-0.875,3.000
RAPID
GOTO/-1,-1.0
CUTTER/2,...,2.8
SPINDL/800
SPINDL/ON,CCLW
RAPID
GOTO/-1,-1.3
GOTO/9.5,3.000
FEADRAT/100,IPM
GOTO/9.5,2.93
GOTO/9.5,2.86
GOTO/9.5,2.79
GOTO/9.5,2.72
GOTO/9.5,2.65
GOTO/9.5,2.58
GOTO/9.5,2.51
GOTO/9.5,2.44
GOTO/9.5,2.37
GOTO/9.5,2.3
GOTO/9.5,2.23
GOTO/9.5,2.16
GOTO/9.5,2.09
GOTO/9.5,2.02
GOTO/9.5,1.95
GOTO/9.5,1.88
GOTO/9.5,1.81
GOTO/9.5,1.74
GOTO/9.5,1.67
GOTO/9.5,1.6
GOTO/9.5,1.53
GOTO/9.5,1.46
GOTO/9.5,1.39
GOTO/9.5,1.32
GOTO/9.5,1.25
GOTO/9.5,1.18
GOTO/9.5,1.11
GOTO/9.5,1.04
GOTO/9.5,0.97
GOTO/9,5,0,9
GOTO/9,5,0,83
GOTO/9,5,0,76
GOTO/9,5,0,69
GOTO/9,5,0,62
GOTO/9,5,0,55
GOTO/9,5,0,48
GOTO/9,5,0,41
GOTO/9,5,0,34
GOTO/9,5,0,27
GOTO/9,5,0,2
RAPID
GOTO/9,5,3,000
RAPID
GOTO/-1,1,-1,3,000
RAPID
GOTO/-1,1,-1,1,0
CUTTER/2,2,1,...,3
SPINDL/800
SPINDL/ON,CCL,SW
RAPID
GOTO/-1,1,-1,1,3
GOTO/5,6,3,000
FEADRAT/100,IPM
GOTO/5,6,2,925
GOTO/5,6,2,85
GOTO/5,6,2,775
GOTO/5,6,2,7
GOTO/5,6,2,625
GOTO/5,6,2,55
GOTO/5,6,2,475
GOTO/5,6,2,4
GOTO/5,6,2,325
GOTO/5,6,2,25
GOTO/5,6,2,175
GOTO/5,6,2,1
GOTO/5,6,2,025
GOTO/5,6,1,95
GOTO/5,6,1,875
GOTO/5,6,1,8
GOTO/5,6,1,725
GOTO/5,6,1,65
GOTO/5,6,1,575
GOTO/5,6,1,5
GOTO/5,6,1,425
GOTO/5,6,1,35
GOTO/5,6,1,275
GOTO/5,6,1,2
GOTO/5,6,1,125
GOTO/5,6,1,05
GOTO/5,6,0,975
GOTO/5,6,0,9
GOTO/5,6,0,825
GOTO/5.6.0.75
GOTO/5.6.0.675
GOTO/5.6.0.6
GOTO/5.6.0.525
GOTO/5.6.0.45
GOTO/5.6.0.375
GOTO/5.6.0.3
GOTO/5.6.0.225
GOTO/5.6.0.15
GOTO/5.6.0.075
GOTO/5.6.0
RAPID
GOTO/5.6.3.000
RAPID
GOTO/-1.1.-1.1.3.000
CUTTER/2,......,5
GOTO/-0.15.-0.15.0
CUTTER/0.3,......,1
RAPID
GOTO/-0.15.-0.15.3
GOTO/1,1.3
FEDRAT/20,IPM
GOTO/1,1.2.95
GOTO/1,1.2.9
GOTO/1,1.2.85
GOTO/1,1.2.8
GOTO/1,1.2.75
GOTO/1,1.2.7
GOTO/1,1.2.65
GOTO/1,1.2.6
GOTO/1,1.2.55
GOTO/1,1.2.5
PAPID
GOTO/1,1.3
RAPID
GOTO/2,2,3.000
FEADRAT/20,IPM
GOTO/2,2,2.95
GOTO/2,2,2.9
GOTO/2,2,2.85
GOTO/2,2,2.8
GOTO/2,2,2.75
GOTO/2,2,2.7
GOTO/2,2,2.65
GOTO/2,2,2.6
GOTO/2,2,2.55
GOTO/2,2,2.5
RAPID
GOTO/2,2,3.000
RAPID
GOTO/1,10,3.000
FEADRAT/20,IPM
GOTO/1,10,2.95
GOTO/1,10,2.9
GOTO/1,10,2.85
GOTO/1,10,2.8
GOTO/1,10,2.75
GOTO/1,10,2.7
GOTO/1,10,2.65
GOTO/1,10,2.6
GOTO/1,10,2.55
GOTO/1,10,2.5
RAPID
GOTO/1,10,3.000
GOTO/-0.15,-0.15,3
GOTO/-0.21,-0.21,0
CUTTER/0.42,.....,1
RAPID
GOTO/-0.21,-0.21,3
GOTO/4,4,3
FEDRAT/20,IPM
GOTO/4,4,2.95
GOTO/4,4,2.9
GOTO/4,4,2.85
GOTO/4,4,2.8
GOTO/4,4,2.75
GOTO/4,4,2.7
GOTO/4,4,2.65
GOTO/4,4,2.6
GOTO/4,4,2.55
GOTO/4,4,2.5
PAPID
GOTO/4,4,3
RAPID
GOTO/10,2,3.000
FEDRAT/20,IPM
GOTO/10,2,2.95
GOTO/10,2,2.9
GOTO/10,2,2.85
GOTO/10,2,2.8
GOTO/10,2,2.75
GOTO/10,2,2.7
GOTO/10,2,2.65
GOTO/10,2,2.6
GOTO/10,2,2.55
GOTO/10,2,2.5
RAPID
GOTO/10,2,3.000
RAPID
GOTO/9,3,3.000
FEDRAT/20,IPM
GOTO/9,3,2.95
GOTO/9,3,2.9
GOTO/9,3,2.85
GOTO/9,3,2.8
GOTO/9,3,2.75
GOTO/9,3,2.7
GOTO/9,3,2.65
GOTO/9,3,2.6
GOTO/9,3,2.55
GOTO/9,3,2.5
RAPID
GOTO/9,3,3.000
RAPID
GOTO/9,2,3.000
FEADRAT/20,IPM
GOTO/9,2,2.95
GOTO/9,2,2.9
GOTO/9,2,2.85
GOTO/9,2,2.8
GOTO/9,2,2.75
GOTO/9,2,2.7
GOTO/9,2,2.65
GOTO/9,2,2.6
GOTO/9,2,2.55
GOTO/9,2,2.5
RAPID
GOTO/9,2,3.000
GOTO/-0.21,-0.21,3
GOTO/-0.3,-0.3,0
CUTTER/0.6,....,1
RAPID
GOTO/-0.3,-0.3,3
GOTO/7,2,3
FEADRAT/20,IPM
GOTO/7,2,2.95
GOTO/7,2,2.9
GOTO/7,2,2.85
GOTO/7,2,2.8
GOTO/7,2,2.75
GOTO/7,2,2.7
GOTO/7,2,2.65
GOTO/7,2,2.6
GOTO/7,2,2.55
GOTO/7,2,2.5
RAPID
GOTO/7,2,3
RAPID
GOTO/2,3,3.000
FEADRAT/20,IPM
GOTO/2,3,2.95
GOTO/2,3,2.9
GOTO/2,3,2.85
GOTO/2,3,2.8
GOTO/2,3,2.75
GOTO/2,3,2.7
GOTO/2,3,2.65
GOTO/2,3,2.6
GOTO/2,3,2.55
GOTO/2.3.2.5
RAPID
GOTO/2.3.3.000
RAPID
GOTO/2.7.3.000
FEADRAT/20,IPM
GOTO/2.7.2.95
GOTO/2.7.2.9
GOTO/2.7.2.85
GOTO/2.7.2.8
GOTO/2.7.2.75
GOTO/2.7.2.7
GOTO/2.7.2.65
GOTO/2.7.2.6
GOTO/2.7.2.55
GOTO/2.7.2.5
RAPID
GOTO/2.7.3.000
RAPID
GOTO/3.8.3.000
FEADRAT/20,IPM
GOTO/3.8.2.95
GOTO/3.8.2.9
GOTO/3.8.2.85
GOTO/3.8.2.8
GOTO/3.8.2.75
GOTO/3.8.2.7
GOTO/3.8.2.65
GOTO/3.8.2.6
GOTO/3.8.2.55
GOTO/3.8.2.5
RAPID
GOTO/3.8.3.000
RAPID
GOTO/7.9.3.000
FEADRAT/20,IPM
GOTO/7.9.2.95
GOTO/7.9.2.9
GOTO/7.9.2.85
GOTO/7.9.2.8
GOTO/7.9.2.75
GOTO/7.9.2.7
GOTO/7.9.2.65
GOTO/7.9.2.6
GOTO/7.9.2.55
GOTO/7.9.2.5
RAPID
GOTO/7.9.3.000
GOTO/-0.3,-0.3,3
GOTO/-0.45,-0.45,0
CUTTER/0.9,.....,1
RAPID
GOTO/-0.45,-0.45,3
GOTO/-1.05,-1.05.3
GOTO/-1.2,-1.2.0
CUTTER/2.4,.....,1
RAPID
GOTO/-1.2,-1.2.3
GOTO/9.5.3
FEDRAT/20,IPM
GOTO/9.5.2.95
GOTO/9.5.2.9
GOTO/9.5.2.85
GOTO/9.5.2.8
GOTO/9.5.2.75
GOTO/9.5.2.7
GOTO/9.5.2.65
GOTO/9.5.2.6
GOTO/9.5.2.55
GOTO/9.5.2.5
PAPID
GOTO/9.5.3
GOTO/-1.2,-1.2.3
GOTO/-1.32,-1.32.0
CUTTER/2.64,.....,1
RAPID
GOTO/-1.32,-1.32.3
GOTO/5.6.3
FEDRAT/20,IPM
GOTO/5.6.2.95
GOTO/5.6.2.9
GOTO/5.6.2.85
GOTO/5.6.2.8
GOTO/5.6.2.75
GOTO/5.6.2.7
GOTO/5.6.2.65
GOTO/5.6.2.6
GOTO/5.6.2.55
GOTO/5.6.2.5
PAPID
GOTO/5.6.3
GOTO/-1.32,-1.32.3
END
VITA AUCTORIS

NAME: Jia Ma

PLACE OF BIRTH Xi’an, P.R. China

YEAR OF BIRTH 1969

EDUCATION Xi’an No.3 High School, P.R. China
1984-1987

Jilin University of Technology, P.R.China