Journal of the Korean Institute
of Industrial Engineers
Vol.20, No.3, September., 1994

A LEARNING SYSTEM BY MODIFYING A DECISION TREE FOR CAPP

이 홍 희*

Hong hee Lee

Abstract

Manufacturing environs constantly change, and any efficient software system to be used in manufacturing must be able to adapt to the varying situations. In a CAPP (Computer-Aided Process Planning) system, a learning capability is necessary for the CAPP system to do change along with the manufacturing system. Unfortunately only a few CAPP systems currently possess learning capabilities. This research aims at the development of a learning system which can increase the knowledge in a CAPP system. A part in the system is represented by frames and described interactively. The process information and process planning logic is represented using a decision tree. The knowledge expansion is carried out through an interactive expansion of the decision tree according to human advice. Algorithms for decision tree modification are developed. A path can be recommended for an unknown part of limited scope. The processes are selected according to the criterion such as minimum time or minimum cost. The decision tree, and the process planning and learning procedures are formally defined.

NOMENCLATURE

$\Rightarrow$: if, then ($A \Rightarrow B$: if A, then B)

$\triangledown$: otherwise ($A \Rightarrow B \triangledown C$: if A,then B, if not A then C.)

$\exists$: there exist(s)

* 인하대학교 산업공학과

¬ : not
∨ : or
∧ : and

1. INTRODUCTION

1.1. An Approach to Learning for Process Planning

Learning is defined as the process in which a human, a living thing, or a computer software system increases or improves its information, knowledge or skill [3]. In low level learning, information or knowledge is increased or improved through a relatively simple process in which acquired data are arranged, compared, or adjusted. In high level learning, information, knowledge, or skill is increased or improved by intelligent reasoning processes such as induction, deduction, etc.

Generally, learning capability is very desirable for an Expert System because its environment is continually changing, and Expert Systems for process planning are no exception. It needs learning capability in order to accept new manufacturing methods, part specifications, and materials which are continually developed and introduced to the manufacturing shop floor.

A CAPP system is not a simple computer program. When a new rule is introduced to the system, it is not an easy task to update the software if its source code is changed. A rule needs to reside in a datafile in order to add or modify it without changing the source code. To accomplish this, the process planning logic is represented using a decision tree, the tree is stored in a datafile, and a learning method is developed based on the decision tree stored in a datafile, in this research.

1.2. Literature on the Learning for CAPP

TOLTEC [4] is one of a few CAPP systems reported in literature which has learning capability. Process planning rules are represented using Production rules. Each Production rule has a certainty value. The certainty value represents the degree of the reliability of a Production rule. TOLTEC has a learning facility of Failure-Driven type. When a certainty value causes an incorrect process plan, the system modifies it interactively.

In Shaw, Menon, and Park's paper [5], learning from observation of an example plan is studied. In their CAPP system, a part is represented by a state. Process information is represented by an operator. In the CAPP system, process planning is carried out by the application of process operators on the state of a part. A typical learning method of Learning from Observation are used as follows: First, the system builds causal relationships between states and operators used. Next, problem-specific parameters are replaced by variables and they are generalized.

2. KNOWLEDGE REPRESENTATION

2.1. Representation of Part Information

In this research, frames describe part information. The frame representation of a part is well studied in several papers [4] [6]. Two kinds of frames are defined here. One is a part frame to describe the general head data such as part name and number, material, principal dimension, and included features. The other is a feature frame to describe the characteristics of each feature of a part such as surface shape, dimensions, dimensional and geometric tolerances, surface roughness, and parent and child features.

2.2. Decision Tree for Learning

Some of the difficulties in implementing of an Expert System are the ordering and consistency checking of its rules and the maintenance of the completeness of its logic. One easy way to avoid the problems is to make a decision tree for reasoning. A decision tree gives a visual advantage for rule ordering and checking of the completeness of the logic. In this research, a decision tree is developed to represent the process planning procedure, and the reasoning will be performed through a decision tree. A learning method of this research relies on the decision tree. The structure of the decision tree is defined for the purpose of learning. A sample decision tree is shown in Figure 1. The upper half of the decision tree represents the classification of a part. It is comprised of a starting node set, a feature node set, feature shape node set, and a material node set. The lower half of the

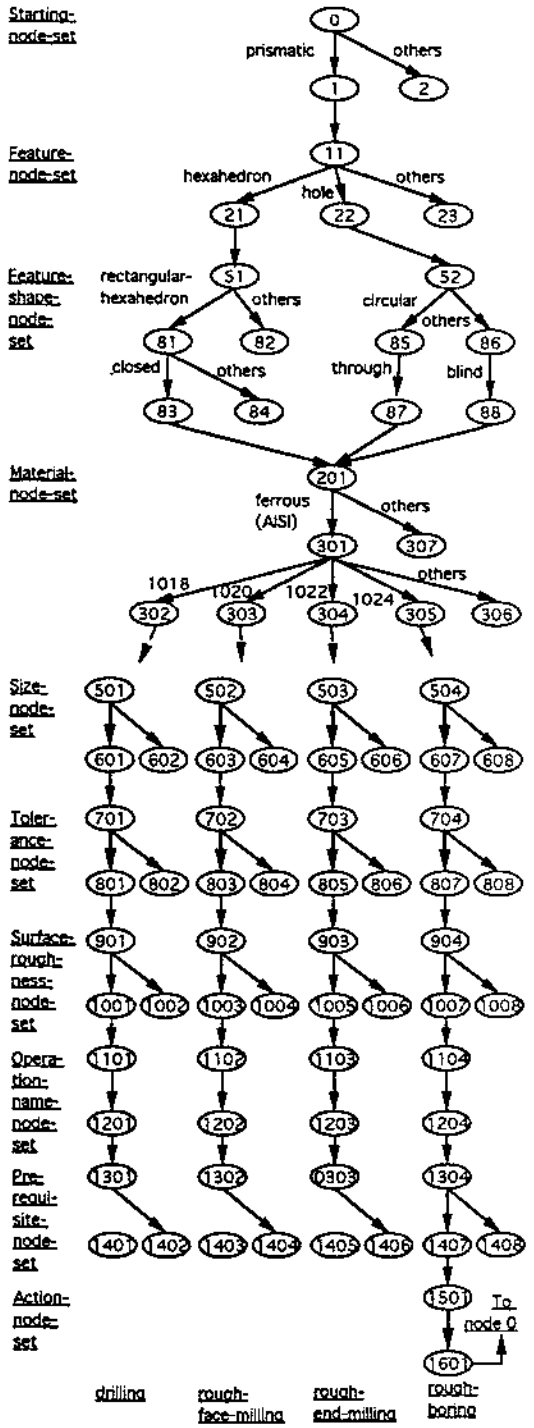


Figure 1. A sample decision tree

decision tree represents the process information. It is comprised of size node sets, tolerance node sets, surface roughness node sets, operation name node sets, prerequisite node set, and action node set.

The basic structure of a decision tree is represented by the following equations:

$$V=\{V_{starting} \cup V_{feature} \cup V_{shape} \cup V_{material} \cup V_{size} \cup V_{tolerance} \cup V_{surface\text{-}roughness} \cup V_{operation\text{-}name} \cup V_{prerequisite} \cup V_{action} \cup V_{cut}\} \quad (1)$$

$$d(V_{starting}) < d(V_{feature}) < d(V_{shape}) < d(V_{material}) < d(V_{size}) < d(V_{tolerance}) < d(V_{surface\text{-}roughness}) < d(V_{operation\text{-}name}) < d(V_{prerequisite}) < d(V_{action}) \quad (2)$$

where d is a depth function which represents the depth of a node, and h is a height function. The depth of a node is the number of distinct nodes included in a path from the starting node in a decision tree. The height is the number of nodes in a node set. A cut node is a dummy node to represent the starting node of each node set. The node sets in a decision tree are comprised as Eq. (1). Eq. (2) defines the depth order of the node sets. In a decision tree, the height of a starting, feature, feature shape, and material node set can be varied. However the height of a size, tolerance, surface roughness, operation name, prerequisite, and action node set can not be varied. In a decision tree, branching is carried out according to the depth order of Eq. (1). The sub-graph from the size node to the action node has only one feasible path, and it is unique to every operation. Each cut node of the node sets from size to action has two output branches: a branch to a terminal node and a vector branch. Several similar attributes make a vector branch and corresponding vector condition.

A decision tree is the main body of a CAPP system. The input of a decision tree is acquired from part and feature frames. The input of a decision tree(IPT), a graph of a decision tree (G),and the output of a decision tree(OPT) are represented as follows:

$$IPT = (PS, FT, FS) \quad (3)$$

where PS is a set of the header information of a part, FT is a set of the features of a part, and FS is a set of the characteristics of each feature.

$$G = (V, E, C, P, L, F) \quad (4)$$

where V is a set of nodes, E is a set of branches, C is a set of conditions, P is a set of paths, L is a set of lengths, and F is a set of functions.

$$OPT = (P_{ST(k)}, L_{ST(k)} \mid k=1,2,\cdots,n_{OPT}) \quad (5)$$

where $P_{ST(k)}$ is the kth path from the starting node to a terminal node and $L_{ST(k)}$ is the length of $P_{ST(k)}$.

3. LEARNING BY TREE EXPANSION

In tree expansion learning, the nodes and branches are added interactively to the existing tree according to the advice of a human. For this advice-taking, it is assumed that all advice is valid. When advice comes into the system, new nodes and branches are generated interactively according to the query format which exists in the system. The examples of the advice is: a specific shape of a feature is made by an operation.

The initial tree of this process planning system contains only the very basic operations such as drilling, rough boring, rough face milling, and rough end milling, which are common to every machining shop. The decision tree is expanded or modified by adding or subtracting nodes and branches. A branch and node in a decision tree is equivalent to a rule. Then, the expansion or modification of a decision tree means the addition or modification of a rule.

3.1. Path Recommending

A CAPP system generates a process plan for a part by classifying the characteristics of a part and comparing the part information and the process information with the information and knowledge of the system. When a characteristic of a part is unknown to the system, the system can not generate a process plan for the part because of the lack of that information. In this case, when other similar characteristics of a part are dealt with by a common procedure of a system, the use of the common procedure for the unknown characteristic can generate an acceptable result. This concept is applied to process planning by a decision tree. In Figure 2, many branches are accumulated to node 18. Then, by connecting a branch from an unknown node, 17, to the common node , 18, a path can be recommended. It can be realized by counting the number of accumulated branches of a candidate output node, and by creating a branch to it, when the number is greater than an specified positive integer.

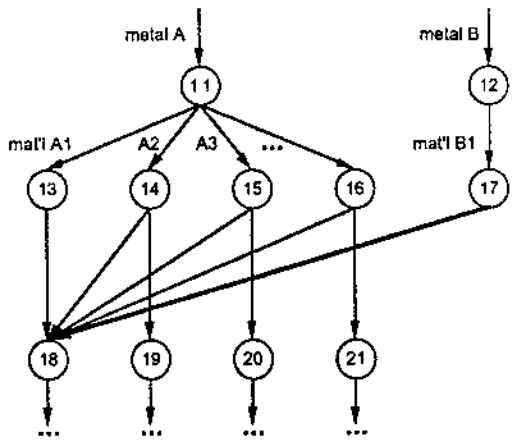


Figure 2. A new branch for an unknown output node

3.2. Learning Functions

A decision tree is expanded by learning functions to augment the knowledge of a system. A learning function set is composed of the creating and deleting functions of a node, an edge, and a condition. The set of learning functions, q, is represented as follows:

$$q = \{q_V, q_E, q_C, s_V, s_E, s_C\}, \quad (6)$$

where q_V is a node creating function, qE is a branch creating function, q_C is a condition creating function, s_V is a node deleting function, sE is a branch deleting function, and s_C is a condition deleting function. Each function is defined as follows:

$$q_v(V_i) = V_i \quad (7)$$

$$qE(V_i, V_j) = E_{ij} \quad (8)$$

$$qC(E_{ij}) = qC(qE(V_i, V_j)) = C_{ij} \quad (9)$$

$$sv(V_i) : deletingV_i \quad (10)$$

$$sE(E_{ij}) = sE(E(V_i, V_j)) : deletingE_{ij} \quad (11)$$

$$sC(C_{ij}) = sC(C(E_{ij})) = sC(C(V_i, V_j)) : deletingC_{ij} \quad (12)$$

3.3. Learning Algorithms

The learning of this research aims at expanding and modifying the process planning knowledge represented using a decision tree. The addition and modification of a rule in a rule-based Expert System is not a simple task because of rule ordering and consistency checking problems. In a decision tree, because each branch and node corresponds to a rule, the above problems can be easily checked visually and the addition and modification of a rule can be done by the expansion and modification of a decision tree.

Five kinds of learning modes are defined to modify a decision tree: Creating a Node and Branch, Creating a Branch, Deleting a Node and Branch, Deleting a Branch, and Path Recommending. Modifying a decision tree can be carried out by the combination of the above five kinds of learning modes. Then, the learning algorithms enable us to add and modify a rule in a decision tree systematically and straightforwardly.

A new graph is formed from an existing graph and an added graph by applying learning functions as follows:

$$q(G, G') = G'' \quad (13)$$

where $G = (V, E, C, P, L, F)$,
$G' = (V', E', C', P', L', F)$, and
$G'' = (V'', E'', C'', P'', LP'', F)$

Each tuple of a new graph is formed by five kinds of algorithms. The nodes of a new graph are formed by applying the mode of creating a node and branch or the mode of deleting a node and branch. The branches and conditions of a new graph are formed similarly by applying the corresponding algorithms. The paths and lengths of a new graph are formed between the nodes of a new graph. This is formally represented by the following equations:

$$V'' = a_{CNB}(V') \vee a_{DNB}(V) \quad (14)$$

$$E'' = aCNB(E') \vee a_{CB}(E') \vee a_{DNB}(E) \vee a_{DB}(E) \vee a_{PR}(E') \quad (15)$$

$$C'' = a_{CNB}(C') \vee a_{CB}(C') \vee a_{DNB}(C) \vee a_{DB}(C) \vee a_{PR}(C') \quad (16)$$

$$P'' = \{P_{ij}\}, V_i V_j \in VP'' \quad (17)$$

$$L'' = \{L_{ij}\}, V_i V_j \in V'' \quad (18)$$

where a_{CNB} is an algorithm to create a node and branch, a_{CB} is an algorithm to create a branch, a_{DNB} is an algorithm to delete a node and branch, a_{DN} is an algorithm to delete a branch, and a_{PR} is an algorithm to create a branch for path recommending.

3.3.1. The Mode of Creating a Node and Branch

In the Mode of Creating a Node and Branch, a node and branch are inserted between corresponding existing nodes, or they are appended to a terminal node in a decision tree. The corresponding branches and conditions are also created, and the old branch and condition are deleted. This mode is formally represented as in Eq. (19). It is explained pictorially in Figure 3 (a) and (b).

$$\begin{aligned}
&\exists V'_j \in V_{(j)} \Rightarrow \\
&\{[\exists V_p V_{(j)} \wedge d(V_p) = d(V'_j)-1] \Rightarrow \\
&[V_p \neq V_{cut|(j)} \Rightarrow V_i = V_p \nabla V_i \in V_{cut|(j)}]\} \\
&\wedge\{[\exists V_p \in V_{(j-1)} \wedge d_{(Vp)} = d(V'_j)-1] \Rightarrow V'j V_{cut|(j)}\} \\
&\wedge\{\exists V_k \wedge d_{(Vk)} = d(V'_j)+1 \Rightarrow \\
&[q_v(V'j) \wedge (qE(V'_j, V_k)) \wedge qC(E(V'_j, V_k) \wedge q_E(V_i, V'_j) \\
&\wedge q_C(E(V_i, V'_j)) \wedge s_E(V_i, V_k) \wedge s_C(E(V_i, V_k))] \\
&\nabla[q_v(V'_{(j)}) \wedge q_E(V_i, V'-(j)) \wedge q_C(E(V_i, V'_{(j)}))\}
\end{aligned} \tag{19}$$

where $V_{(j)|}$ $j=0,1,2,\cdots,9$ is a node set, $V_{cut|(j)}$ is a cut node of $V_{(j)}$, and V'_j is a node which is created in $V_{(j)}$, and V_p is a parant node of V'_j.

3.3.2. The Mode of Creating a Branch

In the Mode of Creating a Branch, a branch is created between two existing nodes in a decision tree. This mode is formally represented in Eq. (20). Eq. (20) is explained pictorially in Figure 3(c).

$$\exists V_i \exists V_j \wedge \neg(\exists E'_{ij}) \Rightarrow q_E(V_i, V_j) \wedge q_C(E(V_i, V_j)) \tag{20}$$

where E'ij is a created or deleted branch.

3.3.3. The Mode of Deleting a Node and Branch .

In the mode of Deleting a Node and Branch, a node and its input branch are deleted. This mode is formally represented in Eq. (21). In Figure 3(d), a non-terminal node and its input and output branches are deleted between two nodes, and in Figure 3(e), a terminal node and its input branch are deleted. The corresponding branches and conditions are also deleted, and the appropriate new branch and condition are created.

$$\begin{aligned}
&\exists V'_j \in V_{(j)} \Rightarrow \\
&\{[\exists V_p \in V_{(j)} \wedge d(V_p) = d(V'_j)-1] \Rightarrow \\
&[Vp \neq V_{cut|(j)} \Rightarrow V_i = V_p \nabla V_i \in V_{cut|(j)}]\} \\
&\wedge\{[\exists V_p \in V_{(j-1)} \wedge d(V_p) = d(V'_j)-1] \Rightarrow V'_j \in V_{cut|(j)}\} \\
&\wedge\{[\exists Vk \wedge d(Vk) = d(V'j)+1] \Rightarrow \\
&[s_v(V'_j) \wedge s_E(E(V_i, V'_j)) \wedge s_C(C(V_i, V'_j)) \\
&s_E(E(V'_j, V_k)) \wedge s_C(C(V'_j, V_k)) \wedge q_E(V_i, V_k) \\
&\wedge q_C(E(V_i, V_k))] \\
&\nabla[s_v(V'_j) \wedge s_E(E(V_i, V'_j)) \wedge s_C(C(V_i, V'_j))]\}
\end{aligned} \tag{21}$$

where V'_j is a deleted.

3.3.4. The Mode of Deleting a Branch

In the Mode of Deleting a Branch, a branch is deleted. When the output node of the deleted branch is a terminal node, it is also deleted.

This mode is formally represented in Eq. (22). Eq. (22) is explained pictorially in Figure 3 (f).

$$[\exists V_i \wedge \exists V_i \wedge \exists E'_{ij}] \{[\neg(\exists V_k) \wedge (d(V_k) = d(V'_j)-1)] \Rightarrow [s_V(V_j) \wedge s_E(V_i,V_j) \wedge s_C(E(V_i,V_j)] \nabla [s_E(V_i,V_j) \wedge s_C(E(V_i,V_j)]\} \quad (22)$$

When a branch from V_i and V_j is deleted, if V_j has another input node other than V_i, E_{ij} is simply deleted. But, if V_j has only one input node V_i, the deletion of E_{ij} means the disconnection of a subtree from V_i. In this case, the disconnected subtree remains as an island subtree. If it is not connected to a main decision tree again, the deletion of E_{ij} is equivalent to erase the subtree which was connected to V_i.

3.3.5. Path Recommending Mode

In Path Recommending Mode, a new branch is created and recommended between two appropriate existing nodes in a decision tree by the learning system. This mode is formally represented in Eq. (23). Eq. (23) is explained pictorially in Figure 3(g). In Eq. (23), THD is a threshold value, i.e., if the number of input nodes for a given node is greater than or equal to THD, then the learning system can recommend a path by connecting a branch to the node.

$$\exists V_i \wedge \exists V_j \wedge [d(V_j) > d(V_i)] \wedge \neg(\exists E'_{ij}) \Rightarrow [i_n(V_j) \geq q_E(V_i,V_j) \wedge q_C(E(V_i,V_j)], \quad (23)$$

where THD is an arbitrary positive integer and i_n is a function of the number of input nodes. Path Recommending mode handles an unknown case by using the most common path - this will not necessarily be correct and must be checked by an experienced planner.

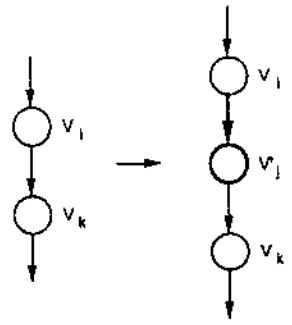
4. PROCESS PLANNING

4.1. Procedure of Process Planning

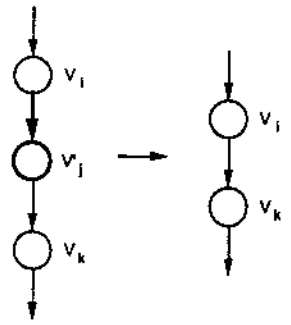
A CAPP system generates a process plan for a mechanical part. First, the system retrieves the necessary information from datafiles. Then, feasible paths are searched in a decision tree and operations are found. During the search, only a few paths are maintained using a Branch and Bound search method according to a minimum machining time criterion. If the searched operation requires a pre-operation, it is determined using recursion. For the recursion, an in-process frame of a pre-feature is made at an action node of a decision tree. The final operation is search first, and the initial operation is search last. For the searched operations, machines are assigned according to minimum machining time or cost. Finally, a process plan is generated.

4.2. CAPP System Developed

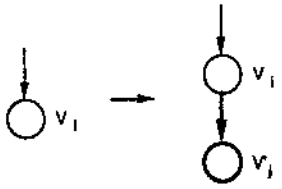
A computer program was developed to demonstrate the concepts and methods developed. The program is composed of three modules and five kinds of datafiles. The modules are Part Input Module, Process



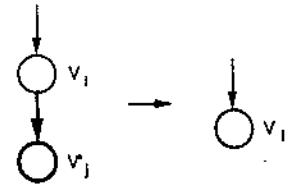
(a) Creating a node and branch



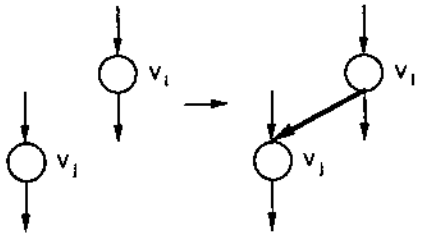
(d) Deleting a node and branch



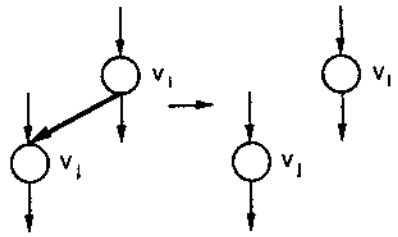
(b) Appending a node and branch to a terminal node



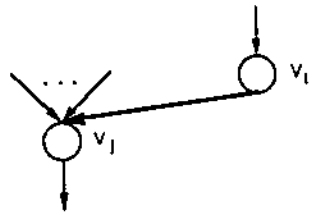
(e) Deleting a terminal node and its branch



(c) Creating a branch between two nodes



(f) Deleting a branch



(g) Recommending a path by creating a branch

Figure 3. Learning modes by tree modifying

Planning Module, and Tree Learning Module. The datafiles are Datafile of Parts, Decision Tree File, Datafile of Processes, Machining Parameter Datafile, and Process Plan File. A part is represented interactively by the Part Input Module, and its datafile is made. Process Planning Module generates a process plan for a part whose information already exists in the datafile using the decision tree data, process data, and machining parameter data in datafiles. This module selects appropriate operations using the decision tree. Machine selection and the evaluation of a process plan are included in this module. It generates a process plan according to the user-specified criterion such as minimum machining time or cost. The generated process plan is stored in Process Plan File. The decision tree data and process data can be expanded interactively by Tree Learning Module. The system was programmed in Common LISP on a Macintosh. The total length of the program is over 12,000 lines or 500K bytes.

5. EXAMPLES

A part of Figure 4 is going to be planned as an example. Apparently, the part can not be planned using the initial decision tree of Figure 1 because of the fine slot and hole. So, the initial decision tree is expanded. For example, the learning is performed as follows:

Node 24, its input branch from node 11, and its condition of Figure 5 are created as

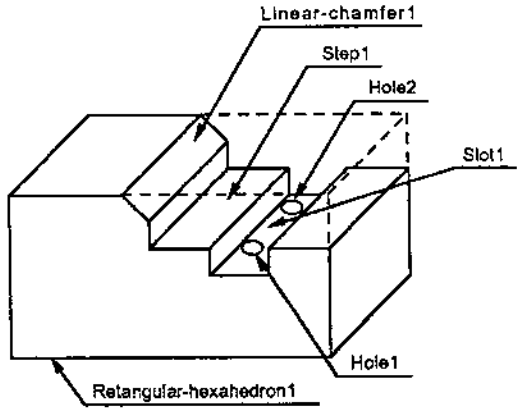


Figure 4. An example part

follows: (1) The user finds node 11 using a tree displaying function of the system. Then, the system shows the output branches of node 11 and their conditions of the current decision tree. Currently, the output nodes of node 11 are nodes 21, 22, and 23, and their conditions are 'hexahedron', 'hole', and 'others' in Figure 1. (2) The user selects Node and Branch Creating Mode. (3) The system asks the condition of the newly creating branch. (4) The user replies as 'step'. (5) The system creates node 24, a branch from node 11 to node 24, and its condition. Then, the current output nodes of node 11 are nodes 21, 22, 23 and 24, and their conditions are 'hexahedron', 'hole', 'others', and 'step'.

This process are repeated using an appropriate tree modifying modes, and the decision tree of Figure 5 is created. Then, the example part is planned using the expanded decision tree. In this example, the material of the part (AISI 1040) is not included in the expanded decision tree. So, the Path Recommending function is

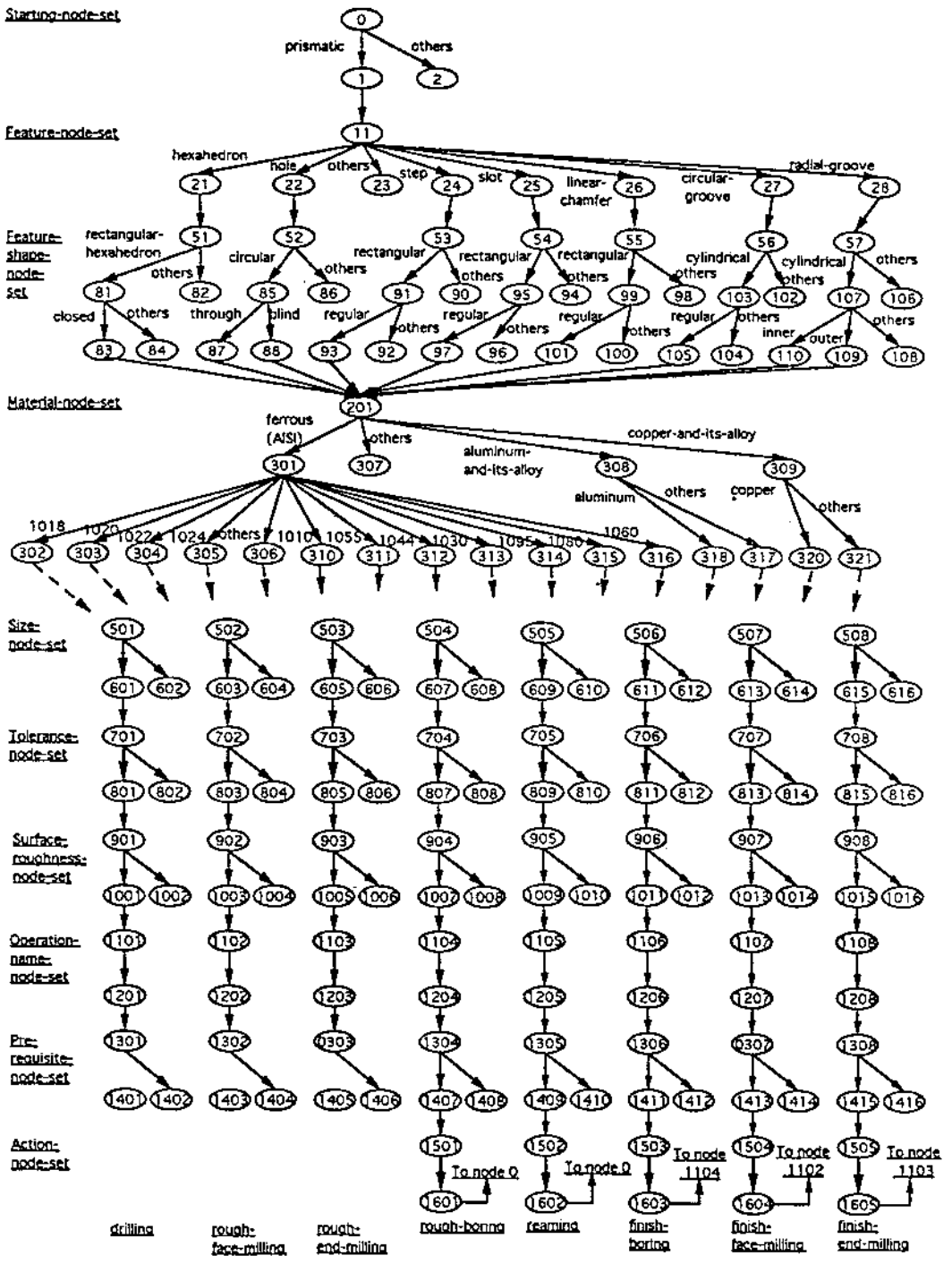


Figure 5. Expanded decision tree

employed. This function finds the subsequent node of AISI1040 node by counting the input nodes of each cut node of size node sets and creating a branch between AISI1040 node and the cut node which has the most input nodes. The generated process plans according to minimum machining cost are in Figure 6. The generated process plan specifies a machine, tool-size, speed, feed, depth of cut, and expected machining time and cost of each feature of a part. Machines are assigned to each feature to reduce the machining costs related to machine tools.

6. CONCLUSIONS

This research was performed in order to develop a learning system for CAPP. In this research, a decision tree represents process planning logic. The developed learning system expands the knowledge of the CAPP system interactively without the modification of the original computer program. Initially, a CAPP system may be developed with only a small amount of process planning knowledge. A sophisticated CAPP system can be easily developed from the initial simple system using the learning system developed. Generally, the development of a rule-based CAPP system is a time-consuming job and requires knowledge engineering. But a process planner can develop a CAPP system by himself by expanding a basic CAPP system in a short time using the learning method developed in this research. A software

```
==========================================
PROCESS PLAN OF :
Part-Name = BLOCKH          Part-Number = 50191
------------------------------------------
[1] Feature: RECTANGULAR-HEXAHEDRON1
        *Operation: ROUGH-FACE-MILLING
        *Machine: VERTICAL-MILLING-MACHINE
    Tool=3.0 Speed=380.0 Feed=0.0096 Depth=0.12
    (Expected-time = 8.82 Expected-cost = 26.49)
[2] Feature: STEP1
        *Operation: ROUGH-END-MILLING
        *Machine: VERTICAL-MILLING-MACHINE
    Tool=2.0 Speed=80.0 Feed=0.004 Depth=0.1
    (Expected-time = 11.29 Expected-cost = 33.91)
[3] Feature: SLOT1
        *Operation: ROUGH-END-MILLING
        *Machine: VERTICAL-MILLING-MACHINE
    Tool=0.75 Speed=80.0 Feed=0.0032 Depth=0.1
    (Expected-time = 2.63 Expected-cost = 7.9)
[4] Feature: SLOT1
        *Operation: FINISH-END-MILLING
        *Machine: VERTICAL-MILLING-MACHINE
    Tool=0.75 Speed=84.0 Feed=0.0024 Depth=0.024
    (Expected-time = 2.03 Expected-cost = 6.1)
[5] Feature: LINEAR-CHAMFER1
        *Operation: ROUGH-END-MILLING
        *Machine: VERTICAL-MILLING-MACHINE
    Tool=0.75 Speed=80.0 Feed=0.0032 Depth=0.1
    (Expected-time = 1.82 Expected-cost = 5.47)
[6] Feature: HOLE1
        *Operation: DRILLING
        *Machine: RADIAL-DRILLING-MACHINE
    Tool=0.875 Speed=68.0 Feed=0.0127 Depth=0.0
    (Expected-time = 1.22 Expected-cost = 1.24)
[7] Feature: HOLE1
        *Operation: REAMING
        *Machine: RADIAL-DRILLING-MACHINE
    Tool=1.0 Speed=40.0 Feed=0.01 Depth=0.0
    (Expected-time = 0.72 Expected-cost = 0.72)
[8] Feature: HOLE2
        *Operation: DRILLING
        *Machine: RADIAL-DRILLING-MACHINE
    Tool=0.875 Speed=68.0 Feed=0.0127 Depth=0.0
    (Expected-time = 0.59 Expected-cost = 0.6)
[9] Feature: HOLE2
        *Operation: REAMING
        *Machine: RADIAL-DRILLING-MACHINE
    Tool=1.0 Speed=40.0 Feed=0.01 Depth=0.0
    (Expected-time = 0.72 Expected-cost = 0.72)
------------------------------------------
Expected-total-time = 29.84
Expected-total-cost = 83.14
(0.0's of some specs represent inapplicable specs.)
(Unit: Tool: inch  Speed: feet/min  Feed: inch/rev
       Depth: inch  Time: min  Cost: dollar )
==========================================
```

Figure 6. Process plan of the example part

system was developed in Common LISP. A CAPP system developed learned new process

planning knowledge and generated a process plan successfully.

REFERENCES

[1] Chang, T.C., and Wysk, R.A., An Introduction to Automated Process Planning Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985

[2] Chen, W.K., Theory of Nets: Flows in Networks, John Wiley & Sons, Inc., New York, 1990

[3] Barr, A., Feigenbaum, E.A., and Cohen, P.R., The Handbook of Artificial Intelligence, Volume I, II, III, Addison-Wesley Publishing Co., Reading, Massachusetts, 1981

[4] Tsatsoulis, C. and Kashyap, R.L., "Using Dynamic Memory Structures in Planning and Its Application to Manufacturing," Purdue University, West Lafayette, Indiana, 1987

[5] Shaw, M.J., Menon, U., and Park, S., "Machine Learning in Knowledge-Based Process Planning Systems," Technical Report, University of Illinois at Urbana-Champaign, College of Engineering, 1989

[6] Nau, D.S., and Chang, T.C., "Prospects for Process Selection Using Artificial Intelligence," *Computers in Industry*, Vol. 4, pp. 253-263, 1983