Journal of the Korean Institute of Industrial Engineers
Vol. 17, No. 2, Dec., 1991

Real Time Scheduling for Computer-Aided Manufacturing (CAM) Systems with Instance-Based Rules

Jong-tae Rhee*

CAM에서의 사례의존규칙을 이용한 실시간 일정계획

이 종 태*

Abstract

An expert scheduling system on real time basis for computer-aided manufacturing systems has been developed. In developing expert scheduling system, the most time-consuming job is to obtain rules from expert schedulers. An efficient process of obtaining rules directly form the schedules produced by expert schedulers is proposed. By the process, a set of complete and minimal set of rules is obtained. During a real time scheduling, when given information on possible values of elements, the rules produce possible values of decision elements, where logical explanations of the result may be offered in terms of chaining rules. The learning and scheduling processes have been simulated with an automated manufacturing line engaged in the production of circuit boards.

1. Introduction

CAM systems are drawing increasing interest because of their capability in responding to various situation such as changes in product demand. The power of those systems, however, is very depending on the efficiency of the real time scheduling technique. In most cases, the problem of real time scheduling in a CAM system is very hard to solve because it cannot be well-formulated in a mathematical programming or other models where a good solution can be found in a reasonable time. Therefore, many systems use fixed rules which does not consider the overall situation of the systems so that the performance of them can be good or very bad in a dynamic situation. Recently, expert

* Purdue University.

scheduling systems(Thesen and Lei, 1986 ; Yih, 1990) was introduced to overcome the difficulty. They have shown the superior performance of an expert scheduling system in a specific domain. With an expert scheduling system, a schedule is automatically selected by a rule chosen from a knowledge base for a specific situation. The performance of an expert system is determined by the quality and reliability of the rules in the knowledge base.

However, the acquisition of rules suffer from the problem of high cost and incompleteness(Nisbett and Wilson, 1977 ; Bainbridge, 1979 ; Ericsson and Simon, 1984 ; Collins, 1985). Conventionally, the acquisition of rules is carried out by interviewing human experts and analyzing collected verbal protocols. Most of the methods require the experts to describe or explain the process of decision making. In some domains, the experts cannot accurately express how they make decisions. The acquiring process may even change experts' view of the problem.

To avoid the difficulty, a different method of rule-acquisition which does not need direct involvement of human experts can be used(Quinlan, 1979 ; Dietterich and Michalski, 1985 ; Kass and Leake, 1987, Rhee and Yih, 1991). That is, the rules are automatically learned from schedules produced by expert schedulers. The set of rules must be complete and non-redundant to minimize cost. In this paper, an expert scheduling system with a process of learning a complete and minimal set of rules is proposed. With the learned rules, each decision making in a real time scheduling can be performed by an individueal rule, or by chaining rules. The process of rule acquisition and decision making has been conducted with an automated manufacturing line engaged in the production of circuit boards.

2. Instance Patterns and Rules

In Fig. 1, a simplified automated line engaged in producing circuit boards is represented. The line consists of chemical process tanks, an input buffer and an output buffer. There is a material handling robot in a track which transports a job among buffers and tanks. Each buffer and tank can contain a job at a time. Given a new job entry in Input buffer, it can be moved to the tanks by a prespecified sequence. To make the problem simple, it is assumed that each circuit board should

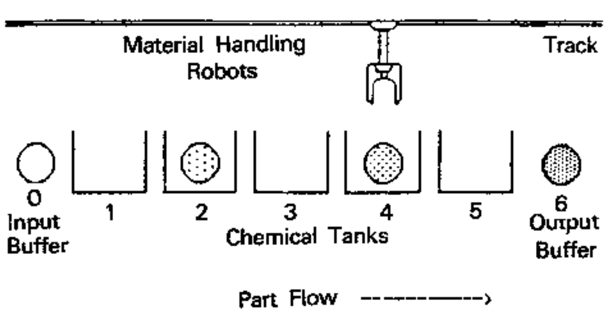

Fig. 1. A simplified circuit boards production line.

pass through the sequence of Tank 1, Tank 2, Tank 3, Tank 4, and Tank 5. Whenever a completed job is moved to Output buffer, it is automatically shipped out. The quality of a produced circuit board is determined by the time of length it spent in each tank. For each circuit board and each tank, a safe time length, $[T^*, T^*+\text{tolerance}]$, is prespecified. ($t^*$ is an optimal time length.) If a circuit board spends within the safe time length in each tank, it becomes as good product. On the other hand, if it stays shorter than the safe time length in a tank, it becomes a bad product. And if it stays longer than the safe time period in a tank, the product will be spoiled. In this research, it is assumed that the optimal time length can be different for each job but the tolerance is set to 24 minutes for all jobs and tanks. The objective of the system is to produce as many good circuit boards as possible in a period. At a time, the status of the system can be represented with the following elements :

$E_0=1$, if Input buffer contains a new part,
$=0$, otherwise.

$E_i=T_i$, if there is not a spoiled job in Tank i, where
T_i is the remained time until safe time length is up for the job in Tank i,
$=0$, if there is no job in Tank i,
$=-1$, if the job in Tank 1 is spoiled,
for $i=1, 2, \cdots 5$.

E_6= the current position of robot(0, 1, 2, …6).

Given a status of the system, the next move of the robot should be determined. The move of the robot can be represented with a couple of elements :

E_7= the position from which the contained job is to be moved (0, 1, 2, …5).

E_8= the position into which the job is to be moved (1, 2, …6).

A real time schedule can be represented as a vector of the elements. That is,

$(E_0, E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8)$

represent a schedule that given a status of the system is by the values of E_0, $E_1, \cdots$, and E_6, the next move of a job is from E_7, to E_8. Such a vector will be called an "instance pattern." In this research, the value range of each E_i, for $i=1, 2, \cdots, 5$, is divided by several ranges and any value in a same range will be regarded as same. That is, for $i=1, 2, 3, 4$, and 5,

$E_i=1$, if $24<T_i$,
$=2$, if $16<T_i<24$,
$=3$, if $8<T_i<16$,
$=4$, if $0<T_i<8$,

and E_i can be 0 and -1, as described above. Note that we can interpret the meaning of the value E_i, $i=1, 2, 3, 4$, and 5,

if $E_i=1$, the job in Tank i is not ready,
if $E_i=2$, the job in Tank i is ready,
if $E_i=3$, the job in Tank i is in hurry, and
if $E_i=4$, the job in Tank i is urgent

for moving out to make a good product. Given a situation, that is, the values of E_0, E_1, E_2, E_3, E_4, E_5, and E_6, a recommendable decision of the next move of the robot, that is, the values of E_7, and E_8, can be described by a rule. For example, we can imagine a rule,

if $E_1=4$, $E_2=0$, and $E_6=1$, $E_7=1$ and $E_8=2$.

Such a rule can be obtained from the schedules of expert schedulers, and then, the rule is called an "instance-based rule." (For simplicity, it will be called just a rule).

From now on, the general concept of instance patterns and rules are explained. In general, let $PATSET=\{PAT_1, PAT_2, \cdots\}$ be a given set of instance patterns and E_i, for $i=1, 2, \cdots, N$, denote the i-th element in an instance pattern where N is the number of elements in each instance pattern. It is assumed that in each instance pattern, the value of each E_i is given. Let $e_i[PAT]$ be the value of E_i in PAT. For example, assuming that $N=4$, if $PAT=(1, 2, 1, 2)$, then $e_1[PAT]=1$, $e_2[PAT]=2$, etc. From PATSET, we can find a relation of values of elements, or a rule, such as

"if $E_1=1$ or 2 and $E_2=2$, then $E_1=1$, $E_2=2$, $E_3=1$ or 2, and $E_4=1$"

Let's denote this rule by $Rule_a$. $Rule_a$ will be represented as

$Rule_a$: $(1 \vee 2, 2, *, *) \rightarrow (1, 2, 1 \vee 2, 1)$,

where "$\vee$" means "or" and "$*$" means "any value of E_i existed in PATSET". For any rule $rule_X$, let $IF(Rule_X)$ and $THEN(Rule_X)$ denote the left part and the right part, or called the "if" part and the "then" part, of $Rule_X$, representing the condition and the conclusion of $Rule_X$. For instance,

$IF(Rule_a)=(1 \vee 2, 2, *, *)$ and

$THEN(Rule_a)=(1, 2, 1 \vee 2, 1)$.

For convenience, "$*$" will be used only in "if" parts of rules (not "then" parts). In this paper, π will denote an "if" part or a "then" part of a rule that can be formed from PATSET and

$V_i(\pi)$

will denote the set of values of E_i assigned in π. For example,

$V_1(IF(Rule_a))=\{1, 2\}$,

$V_2(IF(Rule_a))=\{2\}$,

$V_3(IF(Rule_a))=\{$all the values of E_3 existed in PATSET$\}$, and

$V_4(IF(Rule_a))=\{$all the values of E_4 existed in PATSET$\}$.

Note that for any rule $rule_X$, $V_i(THEN(Rule_X)) \subseteq V_i(IF(Rule_X)), \forall i$. Assume that PATSET is given as follows :

Example 1 : Instance patterns :

$PAT_1=(1, 1, 1, 1)$,

$PAT_2=(1, 1, 2, 2)$,

$PAT_3=(2, 1, 2, 2)$,

$PAT_4=(3, 2, 2, 2)$,

$PAT_5=(3, 3, 2, 3)$,

$PAT_6=(3, 2, 2, 4)$.

Form PATSET, we can form a rule,

$Rule_1$: $(1, *, *, *) \rightarrow$

$(1, 1, 1 \vee 2, 1 \vee)$,

which is interpreted as 'if $E_1=1$, then $E_1=1$, $E_2=1$, $E_3=1$ or 2 and $E_4=1$ or 2" and is observed as such in PATSET because the condition "E_1 is 1" is satisfied only in PAT_1 and PAT_2, $E_1=1$, $E_2=1$, $E_3=1$ or 2, $E_4=1$ or 2. Let

$PATSET[\pi]$

be the subset of PATSET such that for each instance pattern PAT in $PATSET[\pi]$,

$e_i[PAT] \quad V_i[\pi], \forall i$.

As a matter of fact, a rule $Rule_X$ can be found from PATSET if and only if

$PATSET[IF(Rule_X)] \neq \phi$.

Let

$V_i(PATSET[\pi])$

be the set of all the values of E_i existed in $PATSET[\pi]$. (For the purpose of convenience, $V_i(PATSET)$ will denote the set of all the values of E_i given in PATSET.) Then,

$V_i(THEN(Rule_X))=V_i(PATSET[IF(Rule_X)]), \forall i$

Consider the following rule,

$Rule_2$: $(1 \vee 2,\ *,\ *,\ 1 \vee 3 \vee 4) \rightarrow (1,\ 1,\ 1,\ 1)$.

Note that only PAT_1 satisfies $IF(Rule_3)$ so that $THEN(Rule_3) = PAT_1$. Consider the following rule,

$Rule_3$: $(*,\ *,\ *,\ *) \rightarrow (1 \vee 2 \vee 3,\ 1 \vee 2 \vee 3,\ 1 \vee 2,\ 1 \vee 2 \vee 3 \vee 4)$.

Because there are no specified values of elements in $IF(Rule_4)$, $Rule_4$ will be called an "unconditioned rule." Note that $V_i(THEN(Rule_3)) = V_i(PATSET), \vee i$.

Definition : $\pi_1 \leq \pi_2$ if $V_i(\pi_2) \subseteq V_i(\pi_1), \vee i$. Here, if $\pi_1 \neq \pi_2$, $\pi_1 \lneq \pi_2$, which can be interpreted as "π_2 is more specific than π_1."

Ex. $(1,\ 1 \vee 2,\ 1 \vee 2,\ 1 \vee 2 \vee 3) \angle (1,\ 1,\ 1 \vee 2,\ 1)$.

Note that for any rule $Rule_X$, $IF(Rule_X) \lneq THEN(Rule_X)$.

3. Complete and Minimal Set of Rules

From a set of instance patterns, PATSET, we can form many rules but they can be redundant. For example, consider the following rules,

$Rule_4$: $(1 \vee 2,\ *,\ *,\ *) \rightarrow (1 \vee 2,\ 1,\ 1 \vee 2,\ 1 \vee 2)$,

$Rule_5$: $(*,\ 1 \vee 2,\ *,\ 2 \vee 3) \rightarrow (1 \vee 2 \vee 3,\ 1 \vee 2,\ 2,\ 2)$ and

$Rule_6$: $(1 \vee 2,\ *,\ *,\ 2 \vee 3 \vee 4) \rightarrow (1 \vee 2,\ 1,\ 2,\ 2)$,

formed from PATSET in Example 1. $Rule_6$ is redundant with $Rule_4$ and $Rule_5$. Let an initial state of elements is given as represented in $IF(Rule_6)$. That is,

$E_1 = 1$ or 2, and $E_4 = 2$ or 3 or 4.

Then, $IF(Rule_4)$ is satisfied by the state so that from $THEN(Rule_4)$,

$E_1 = 1$ or 2, $E_2 = 1$ $E_3 = 1$ or 2 and $E_4 = 1$ or 2,

and therefore, the state of elements is updated as follows :

$E_1 = 1$ or 2, $E_2 = 1$ $E_3 = 1$ or 2 and $E_4 = 2$.

Now, $IF(Rule_5)$ is satisfied by the state (and the state of elements will be updated again by $THEN(Rule_5)$). Note that

$V_i(THEN(Rule_6)) = V_i(THEN(Rule_4)) \cap V_i(THEN(Rule_5)), \forall i$.

That is, if the state of elements is given as $IF(Rule_6)$, then $IF(Rule_4)$ and $IF(Rule_5)$ are satisfied "in chain" and the conclusion given in $THEN(Rule_6)$ can be produced by combining $THEN(Rule_4)$ and $THEN(Rule_5)$.

Definition : $Rule_X$ "can be explained by chaining" $Rule_a$, $Rule_b$, $\cdots$, and $Rule_t$ if, an initial state of elements is given as represented in $IF(Rule_X)$,

(I) if the initial state satisfies $IF(Rule_a)$, that is,

$V_i(IF(Rule_X)) \subseteq V_i(IF(Rule_a)), \forall i$,

the updated state of elements by $THEN(Rule_a)$ satisfies $IF(Rule_b)$, that is,

$V_i(IF(Rule_X)) \cap V_i(THEN(Rule_a)) \subseteq V_i(IF(Rule_b)), \forall i$,

the updated state of elements by $THEN(Rule_b)$ satisfies $IF(Rule_c)$, that is,

$V_i(IF(Rule_X)) \cap V_i(THEN(Rule_a)) \cap V_i(THEN(Rule_b))$

$\subseteq V_i(IF(Rule_c)), \forall i$,

$\cdots$, and the updated state of elements by $THEN(Rule_s)$ satisfies $IF(Rule_t)$, that is,

$V_i(IF(Rule_X)) \cap V_i(THEN(Rule_a)) \cap V_i(THEN(Rule_b)) \cdots \cap V_i(THEN(Rule_s))$

$\subseteq V_i(IF(Rule_i)), \forall i,$

and

(II) $V_i(THEN(Rule_X))$

$= V_i(THEN(Rule_a)) \cap V_i(THEN(Rule_b)) \cap \cdots \cap V_i(THEN(Rule_t)), \forall i.$

Note that a rule $Rule_X$ "can be explained by a single rule" $Rule_Y$, if

$IF(Rule_Y) \angle IF(Rule_X)$ and $THEN(Rule_X) = THEN(Rule_Y)$.

Let ALL(PATSET) be the set of all the rules that can be obtained from PATSET.

Definition : A set of rules obtainable from PATSET is defined as a complete and minimal set of rules, represented as MIN(PATSET), if

i) each rule in ALL(PATSET)−MIN(PATSET) can be explained by chaining rules in MIN(PATSET) and

ii) each rule in MIN(PATSET) cannot be explained by chaining other rules in MIN(PATSET).

Note that MIN(PATSET) is a non-redundant set of rules and any rule that can be formed from PATSET is a rule of MIN(PATSET) or can be explained by chaining rules of MIN(PATSET)

4. Learning Algorithm of a complete and minimal set of rules

Let COM(PATSET) be the set of rules obtainable from PATSET such that

i) each rule in ALL(PATSET)−COM(PATSET) can be explained by a single rule in MIN(PATSET) and

ii) each rule in COM(PATSET) cannot be explained by a single rule in MIN(PATSET).

Note that MIN(PATSET) is a subset of COM(PATSET). Let PAT_i, i=1, 2,⋯ be the i^{th} instance pattern presented and $PATSET_i = \{PAT_1, PAT_2, \cdots, PAT_i\}$.

ALGORITHM

Phase 1. Initialize. Obtain COM($PATSET_1$).

For p=2, 3,⋯, up to the fnal instance pattern PATSET, do the following :

Phase p. Obtain COM($PATSET_p$) from COM($PATSET_{p-1}$) and PAT_p.

p-1 : Find and modify incorrect rules in COM($PATSET_{p-1}$).

p-2 : Create new rules.

p-3 : Obtain each rule that can be explained by another single rule in COM($PATSET_{p-1}$) which was found incorrect in p-1.

p-4 : From the rules in hand, obtain COM($PATSET_p$).

Phase*. From the rules obtained, obtain MIN(PATSET).

In Phase 1, with the first instance pattern, PAT_1, COM($PATSET_1$) is easily obtained.

Actually, it has one and only one rule, $Rule_X$, such that

$IF(Rule_X = (*, *, \cdots, *)$ and

$THEN(Rule_X) = PAT_1$.

For example, if $PAT_1 = (1, 1, 1, 1)$, the only rule in COM($PATSET_1$) is

$Rule_X : (*, *, *, *) \rightarrow (1, 1, 1, 1)$

because for any other rule $Rule_Y$ in ALL(PAT_1),

$IF(Rule_X) \angle IF(Rule_Y)$ and $THEN(Rule_Y) = THEN(Rule_X)$.

That is, $Rule_Y$ can be explained by $Rule_X$.

Consider Phase p-1. Let $Rule_X°$ be a rule in ALL

$(PATSET_{p-1})$ and $Rule_X$ be the corresponding rule in $ALL(PATSET_p)$ such that $Rule_X^o$ is modified to $Rule_X$ according to PAT_p. For any rule, whether it is in $ALL(PATSET_{p-1})$ or in $ALL(PATSET_p)$ will be represented by whether "o" exists or not in the name of the rule. For convenience, let

$e_i^a = e_i[PAT_p], \forall i.$

Definition : $Rule_X^o$ is incorrect with PAT_p in E_m if PAT_p satisfies $IF(Rule_X^o)$ and $e_i^a \notin V_m(THEN(Rule_X^o))$.

$Rule_X^o$ is modified to $Rule_X$ as follows :

$IF(Rule_X) = IF(Rule_X^o)$ and

$V_i(THEN(Rule_X)) = V_i(THEN(Rule_X^o)) \cup \{e_i^a\}, \forall i.$

For example, assume that we have

$Rule_X^o : (*, *, *, *) \rightarrow (1, 1, 1, 1)$ and

$PAT_p = (1, 1, 2, 2)$.

$Rule_X^o$ is incorrect with PAT_p in E_2 and E_3 so that $Rule_X^o$ must be modified to

$Rule_X : (*, *, *, *) \rightarrow (1, 1, 1 \vee 2, 1, \vee 2)$.

Note that if $Rule_X^o$ is correct with PAT_p, $Rule_X = Rule_X^o$.

Note that in phase p-1, we do not find and modify incorrect rule in the set of $ALL(PATSET_{p-1})$-$COM(PATSET_{p-1})$. The reason is as follows :

Assume the $Rule_Y^o$ is a rule in the above set and is incorrect with PAT_p in E_m. Then, $Rule_Y^o$ can be explained by another rule $Rule_x^o$ in $COM(PATSET_{p-1})$. Because

$IF(Rule_X^o) \angle IF(Rule_Y^o)$ and $THEN(Rule_Y^o) = THEN(Rule_Y^o)$,

$Rule_x^o$ must be incorrect with PAT_p in E_m, too. Therefore,

$THEN(Rule_Y) = THEN(Rule_X)$.

And because

$IF(Rule_X) = IF(Rule_X^o) \angle IF(Rule_Y) = IF(Rule_Y^o)$,

$Rule_Y$ can be explained by $Rule_X$. Therefore, $Rule_Y$ cannot be in $COM(PATSET_p)$ so that it need not be obtained.

In Phase p-2, consider new rule $Rule_X$, that is, there does not exist a corresponding rule $Rule_X^o$. Then $PATSET_{p-1}(IF(Rule_X))$ must be empty because if not, $Rule_X^o$ exists. Therefore $PATSET_p(IF(Rule_X)) = \{PAT_p\}$ so that

$THEN(Rule_X) = PAT_p$.

Note that if there is another new rule $Rule_Y$ such that $IF(Rule_Y) \angle IF(Rule_X)$, and $THEN(Rule_Y) = PAT_p$, $Rule_X$ can be explained by $Rule_Y$ so that $Rule_X$ cannot be in $MIN(PATSET_p)$. Therefore, if $Rule_X$ is a rule of $MIN(PATSET_p)$, it must be that for each element E_m such that $V_m(IF(Rule_X)) \subset V_m(PATSET_p)$ and for any rule $Rule_Z$ such that

$V_i(IF(Rule_X)) = V_i(IF(Rule_Z))$ for $i \neq m$ and

$V_m(IF(Rule_X)) \subset V_m(IF(Rule_Z))$,

$Rule_Z$ is not a new rule, that is, $Rule_Z^o$ exists. Note that PAT_p satisfies $IF(Rule_Z^o)$. Also note that $Rule_Z^o$ must be incorrect with PAT_p in E_m, that is, $e_m^a \notin V_m(THEN(Rule_Z^o))$. It is because if $e_m^a \in V_m(THEN(Rule_Z^o))$, $PATSET_{p-1}(IF(Rule_Z^o))$ must have an instance pattern PAT where $e_m[PAT] = e_m^a$ and because

$IF(Rule_Z) = IF(Rule_Z^o)$ and $e_m^a \in V_m(IF(Rule_X))$,

PAT will satisfy $IF(Rule_X)$ and therefore, $PATSET_{p-1}(IF(Rule_X))$ will not be empty. And for each value e_m^* of E_m such that $e_m^* \in V_m(THEN(Rule_Z^o))$, it must be that

$e_m^* \notin V_m(IF(Rule_X))$

because $PATSET_{p-1}(IF(Rule_Z^o))$ has an instance pattern PAT where $e_m PAT] = e_m^*$, so that if $e_m^* \in V_m(IF(Rule_X))$, PAT will satisfy $IF(Rule_X)$ and therefore,

$PATSET_{p-1}(IF(Rule_X)) \neq \phi$

A new rule $Rule_X$ can be constructed from each rule $Rule_Z°$ in $ALL(PATSET_{p-1})$ and each element E_m such that $Rule_Z°$ is incorrect with PAT_p:

$$V_i(IF(Rule_X))=V_i(IF(Rule_Z°)) \text{ for } i\neq m, \text{ and} \quad \cdots(6)$$

$$e_m^{\alpha}\in V_m(IF(Rule_X)) \text{ and } V_m(IF(Rule_X))\cap V_m(THEN(Rule_Z°))=\phi \quad \cdots(7)$$

Note that there could be multiple rules that satisfy (7). Among them, we only have to obtain $Rule_X^*$ such that

$$V_m(IF(Rule_X^*))=\{e_i^{\alpha}\}\cap(V_m(PATSET_{p-1})-V_m(THEN(Rule_Z°)) \quad \cdots(8)$$

because all other rules will have more specific "if" part than $IF(Rule_X^*)$ so that they can be explained by $Rule_X^*$. From now on $Rule_X^*$ will be represented by $(Rule_Z°)_m^{\alpha}$.

Theorem : In Phase p-2, we only have to generate $(Rule_W°)_m^{\alpha}$ for each rule $Rule_W^{\alpha}$ and E_m such that $Rule_W°$ is incorrect with PAT_p in E_m and $Rule_W°$ is in $COM(PATSET_{p-1})$.

proof Consider a rule $Rule_Z°$ which is in $ALL(PATSET_{p-1})-COM(PATSET_{p-1})$. Then $Rule_Z°$ can be explained by a rule $Rule_W°$ in $COM(PATSET_{p-1})$. Assume that $Rule_Z°$ is incorrect with PAT_p in an element E_m. Then, $Rule_W°$ is also incorrect with PAT_p in E_m so that we can consider $(Rule_W°)_m^{\alpha}$ and $(Rule_Z°)_m^{\alpha}$. Because the "if" part of $(Rule_W°)_m^{\alpha}$ is less specific than the "if" part of $(Rule_Z°_m^{\alpha}$ and the "then" parts of $(Rule_W°)_m^{\alpha}$ and $(Rule_Z°)_m^{\alpha}$ are the same, $(Rule_Z°)_m^{\alpha}$ can be explained by $(Rule_W°)_m^{\alpha}$. Therefore, $(Rule_Z°)_m^{\alpha}$ cannot be in $COM(PATSET_p)$ and need not be generated.

Let $Rule_U°$ be the unconditioned rule formed from $PATSET_{p-1}$. Note that $Rule_U°$ must be in $COM(PATSET_{p-1})$, and $V_m(PATSET_{p-1})$ can be found in $THEN(Rule_U°)$ because $V_m(THEN(Rule_U°))=V_m(PATSET_{p-1}))$. For example, assume that we have

$Rule_U°$: $(*, *, *, *)\rightarrow(1\vee 2\vee 3, 1\vee 2, 1\vee 2\vee 3, 1\vee 2)$,

$Rule_W°$: $(1\vee 2, *, 1\vee 2, *)\rightarrow(1, 1, 1, 1\vee 2)$ and

$PAT_p=(1, 2, 2, 2)$.

$Rule_W°$ is incorrect with PAT_p in E_2 and E_3 and new rules to be generated corresponding to E_2 and E_3 are

$(Rule_W°)_2^{\alpha}$: $(1\vee 2, 2, 1\vee 2, *)\rightarrow(1, 2, 2, 2)$ and

$(Rule_W°)_3^{\alpha}$: $(1\vee 2, *, 2\vee 3, *)\rightarrow(1, 2, 2, 2)$

Now, consider Phase p-3. Let $Rule_X°$ can be explained by a rule $Rule_Y°$ in $COM(PATSET_{p-1})$. As defined above, let $e_i\alpha=e_i[PAT_p]$, $\vee$ i.

Lemma : If $Rule_X°$ can be explained by $Rule_Y°$ and if $Rule_X$ is a rule in $COM(PATSET_p)$, (A) there is an element E_m such that $Rule_Y°$ is incorrect with PAT_p in E_m but $Rule_X°$ is not incorrect with PAT_p in E_m,

(B) $IF(Rule_X°)=IF(Rule_Y°)$ except that $e_a^{\alpha}\in V_i(IF(Rule_Y°))$ but $e_a^{\alpha}\notin V_i(IF(Rule_X°))$ for one and only one element E_a such that $Rule_Y°$ is incorrect with PAT_p in E_a but $Rule_X°$ is not incorrect with PAT_p in E_a

proof of A) Assume that there is no such element, that is, for each element E_i such that $Rule_Y°$ is incorrect with PAT_p in E_i, $Rule_X°$ is also incorrect with PAT_p in E_i. Note that because $IF(Rule_Y°)<IF(Rule_X°)$ and $THEN(Rule_X°)=THEN(Rule_Y$

°), if $Rule_X°$ is incorrect with PAT_p in an element E_j, $Rule_Y°$ is also incorrect with PAT_p in E_j. Therefore, THEN($Rule_X$) = THEN($Rule_Y$). And because IF($Rule_Y$) = IF($Rule_Y°$) ∠ IF($Rule_X°$) = IF($Rule_X$), $Rule_X$ can be explained by $Rule_Y$, and therefore, $Rule_X$ should not be in COM($PATSET_p$).

proof of B) Note that the above conclusion (A) implies that IF($Rule_X°$) is not satisfied by PAT_p because if satisfied, THEN($Rule_X°$) = THEN($Rule_Y°$) and $Rule_X°$ is incorrect with PAT_p in E_m. (therefore, $Rule_X°$ must be correct with PAT_p). Then, there must exist at least one element E_a such that

$$e_a^{\alpha} \in V_a(Rule_Y°)) \text{ but } e_a^{\alpha} \notin (IF(Rule_X°)). \quad \cdots\cdots (1)$$

consider $Rule_Z°$ such that IF($Rule_Z°$) = IF($Rule_Y°$) except that $e_a^{\alpha} \in V_a(IF(Rule_Y°))$ but $e_a^{\alpha} \notin V_a$ (IF(IF$Rule_Z°$)).

Assume that there is another element F_b such that

$e_b^{\alpha} \in V_b(IF(Rule_Y°))$ but $e_b^{\alpha} \notin V_b(IF(Rule_X°))$.

Then, IF($Rule_Y°$) ∠ IF($Rule_Z°$) ∠ IF($Rule_X°$). Because THEN($Rule_X°$) = THEN($Rule_Y°$), it must be that THEN($Rule_Z°$) = THEN($Rule_X°$). Therefore, $Rule_X°$ can be explained by $Rule_Z°$. Furthermore, because both rules are correct with PAT_p, $Rule_X$ can be explained by $Rule_Z$ and $Rule_X$ should not be in MIN($PATSET_p$). Therefore, in this case, $Rule_X°$ does not need to be restored. Therefore, there must be one and only one element E_a such that (1) holds. From (10), $e_a^{\alpha} \notin V_a(IF(Rule_X°))$ and IF($Rule_X°$) $\not\angle$ THEN($Rule_X°$), so

$e_a^{\alpha} \notin V_a(THEN(Rule_X°)) = V_a(THEN(Rule_Y°))$.

And because $e_a^{\alpha} \in V_a(IF(Rule_Y°))$, $Rule_Y°$ must be incorrect with PAT_p in E_a.

The above lemma implies that in Phase p-3, for each rule $Rule_Y°$ that is a rule in COM($PATSET_{p-1}$) and for each element E_m such that $Rule_Y°$ is incorrect with PAT_p in E_m, we need to restore the rule $Rule_X°$ as follows.

IF($Rule_X°$) = IF($Rule_Y°$) except that $e_m^{\alpha} \in V_m$(IF ($Rule_Y°$)) but $e_m^{\alpha} \notin V_m(IF(Rule_X°))$,

(that is, $V_m(IF(Rule_X°)) = V_m(IF(Rule_Y°)) - \{e_m\alpha\}$), and

THEN($Rule_X°$) = THEN($Rule_Y°$).

(Note that the rule $Rule_Y°$ is either in MIN($PATSET_{p-1}$) or is restored in Phase 1-1). Here, it will be represented by

$Rule_X° = (Rule_Y°)_m^{\beta}$.

Note that in the case of $V_m(Rule_Y°)) = V_m(PATSET_{p-1})$, $V_m(IF(Rule_Y°))$ can be found from V_m (THEN($Rule_U°$)). For example, assume that we have

$Rule_U°$: (*, *, *, *) → (1 ∨ 2 ∨ 3, 1 ∨ 2, 1 ∨ 2 ∨ 3, 1 ∨ 2),

$Rule_Y°$: (1 ∨ 2 *, 1 ∨ 2, *) → (1, 1, 1, 1 ∨ 2) and

PAT_p = (1, 2, 2, 2).

$Rule_Y°$ is incorrect with PAT_p in E_2 and E_3 and rules to be restored corresponding to E_2 and E_3 are

$(Rule_Y°)_2^{\beta}$: (1 ∨ 2, 1, 1 ∨ 2, *) → (1, 1, 1, 1 ∨ 2) and

$(Rule_Y°)_3^{\beta}$: (1 ∨ 2, *, 1, *) → (1, 1, 1, 1 ∨ 2).

Phase p-4 can be done as follows : Check each rule, $Rule_X$, on hand in any sequence if $Rule_X$ can be explained by another rule on hand. If it is, delete $Rule_X$.

Phase* can be done as follows : Check each rule,

$Rule_X$, on hand in any sequence if $Rule_X$ can be explained by chaining rules remained on hand. It can be done as follows :

Set an initial state as represented in IF($Rule_x$) and find another rule, $Rule_Y$, such that IF($Rule_Y$) is satisfied by the state and update the state with THEN($Rule_Y$) and similarly, find another rule whose "if" part is satisfied by the updated state and update the state again, etc. until there is no other rules whose "if" part is satisfied by the updated state. And then, we can check if $Rule_X$ can be explained by chaining those activated rules as described in the definition.

5. Experiment

The real time scheduling of the manufacturing line engaged in the producing circuit boards, as described before, was performed by over 100 students. A student was selected as an expert scheduler based on performance. From the data of the schedules produced by the expert scheduler, a complete and minimal set of rules was obtained by the algorithm suggested in the previous section. The number of schedules made by the expert scheduler for obtaining rules was 2125 and the number of learned rules of a complete and minimal set was 176. In other words, the behavior of the expert scheduler can be explained by the 176 rules. Examples of learned rules are listed below :

1) If $E_1=4$, $E_2=0$, and E_3, E_4, $E_5=0$, 1, 2 or 3, then $E_7=1$, $E_8=2$
2) If E_1, E_2, E_3, E_4, $=0$, 1, 2, and $E_5=2$, 3 or 4, then $E_7=5$, $E_8=6$
3) If $E_1=0$, or 1, $E_2=3$ or 4, $E_3=0$, $E_4=3$, and $E_5=0$, 1, 2, then $E_7=2$, $E_8=3$
4) If $E_1=-1$, and E_2, E_3, E_4, $E_5=0$, 1, or 2, then $E_7=1$, $E_8=6$
5) If $E_1=0$, 1, 2, or 3, E_2, $E_3=1$, 2, 3, or 4, $E_4=1$, 2, 3, or 4 and $E_5=0$, then $E_7=4$, $E_8=5$
6) If $E_1=0$, or 1, $E_2=0$, or 1, $E_3=-1$, $E_4=0$, and $E_5=0$, or 1, $E_7=3$, $E_8=4$
7) If E_1, $E_2=1$ or 2, $E_3=4$, $E_4=1$ or 2, and $E_5=0$, then $E_7=4$, $E_8=5$
8) If $E_1=0$ or 1, $E_2=2$ or 3, $E_3=0$ $E_4=-1$ and $E_5=0$, 1, or 2, then $E_7=2$ $E_8=3$
9) If $E_0=1$, E_2, E_3, E_4, $E_5=0$ or 1, then $E_7=0$, $E_8=1$
10) If $E_1=1$ or 2, $E_2=0$, $E_3=0$ or 1n $E_4=0$ or 1, $E_5=0$, and $E_6=0$, 1, or 2, then $E_7=2$ $E_8=3$

The learned rules agree with the common sense even though some are more or less technical. For example, the rule of 1) represents the behavior of the expert scheduler that if the job in Tank 1 is urgent to be moved out, there is no job in Tank 2 (so that a job can be moved into Tank 2), and the other jobs in the other tanks are not urgent to be moved out, the next move of the robot is to move the job in Tank 1 to tank 2. The rules 4) represents that if the job in Tank 1 is spoiled and the other jobs in the other tanks are not in hurry or urgent to be moved out, the next move of the robot is to take the job in Tank 1 away to Output buffer. The followings are some cases of schedules generated by the learned rules with random situations :

1) Situation : $E_0=0$, $E_1=1$, $E_2=2$, $E_3=0$, $E_4=0$, $E_5=1$, $E_6=5$

Generated schedule : $E_7=2$, $E_8=3$

2) Situation : $E_0=1$, $E_1=0$, $E_2=1$, $E_3=1$, $E_4=0$, $E_5=1$, $E_6=2$

Generated schedule : $E_7=0$, $E_8=1$

3) Situation : $E_0=0$, $E_1=0$, $E_2=0$, $E_3=0$, $E_4=-1$, $E_5=2$, $E_6=4$

Generated schedule : $E_7=5$, $E_8=6$

4) Situation : $E_0=0$, $E_1=1$, $E_2=1$, $E_3=1$, $E_4=2$, $E_5=0$, $E_6=1$

Generated schedule : $E_7=4$, $E_8=5$

5) Situation : $E_0=1$, $E_1=0$, $E_2=1$, $E_3=1$, $E_4=-1$, $E_5=0$, $E_6=2$

Generated schedule : $E_7=4$, $E_8=6$

6. conclusion

In this paper, an expert scheduling system where rules for the knowledge base are obtained from schedules generated by expert scheudlers was suggested. A case of computer-aided manufacturing line producing circuit boards was adopted for simulation. By the rule-generation method, the behavior of an expert scheduler in decision making was saved in a complete and minimal set of rules to be used in later real time scheduling.

Obviously, the learning algorithm offers a great advantage of bypassing the troublesome and costly procedure of obtaining rules from a domain expert. The quality of the obtained rules is closely related to the quality of the schedules used for obtaining those rules and how well the elements are set up. In some problem domains, a probabilistic conclusion is more realistic. The suggested system can be extended to the case of frequency-based probabilistic rules. This remains for future research.

References

[1] Bainbridge, L., Verbal reports as evidence of the process operator's knowledge, *International Journal of Man-Machine Studies*, 11, pp. 411-436, 1979.

[2] Collins, H. M., *Changing Order : Replication and Induction in Scientific Practice.*, London : Sage, 1985.

[3] Dietterich, T. G. and Michalski, R. S., discovering Patterns in Sequence of Events, *Artificial Intelligence* 25, pp. 187-232, 1985.

[4] Ericsson, K. A. & Simon, H. A., *Protocol Analysis : Verbal reports as Data.*, Massachusetts : The MIT Press, 1984.

[5] Kass, Alex M. and Leake, David B., A Case-Based Approach to Building explanations for Explanation-based Learning, Submitted to *AAAI*-87.

[6] Kolodner, J. L., Simpson, R. L. Jr., and Sycara-Cyranski, K., A Process Model of Cased-Based Reasoning in Problem Solving, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 284-290, 1985.

[7] Mitchell, T. M., Version Spaces : A Candidate Elimination Approach to Rule Learning, *Proceedings of the fifth International Joint Conference on Artificial Intelligence*, Los Altos : CA, Morgan Kaufmann, 1977.

[8] Nisbett, R. E., & Wilson, T. D., telling more than we can know : verbal reports on mental processes. *Psyschological Review*, 84, pp. 231-259, 1977.

[9] Quinlan, J. R., Discovering rules by induction from large collections of examples, *Expert System in the Microelectronic Age*, pp. 168-201., Mi-

chie, D.(Ed), edinburgh : Edinburgh University Press, 1979.

[10] J. Rhee and Y. Yih, Instance-Based Inference Rule Learning Algorithm-A Binary Case, Technical Report 91-7, School of Industrial Engineering, Purdue University, West Lafayette, IN, 1991.

[11] Thesen, A. and Lei, L., An expert system for Scheduling Robots in Flexible Electo\roplating System with Dynamically changing Workloads, in *Proceedings of the Second ORSA/TIMS Conference* on FMS : Operaions Research Models and Applications, pp.555-66.

[12] Yih, Y., Trace-driven Knowledge Acquistion (TDKA) for rule-Based Real Time Scheduling Systems, *Journal of Intelligent Manufacturing* 1, pp.217-230, 1990.