An Efficient Method of Solving Problems of Classification and Selection Using Minimum Spanning Tree in a Flexible Manufacturing System

Pei-Sen Liu and Li-Chen Fu

Department of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

Flexible manufacturing systems (FMS's) become more and more popular nowadays in the field of manufacturing since they can perform different workpieces automatically in more than one sequence and, hence, possess considerable flexibility. Design of an FMS will involve several levels of consideration and each level will endeavor to solve various problems. One of those levels is planning and scheduling where problems regarding how to perform *classification* and *selection* are intrinsic ones. In this paper, an efficient heuristic method by building some minimum spanning tree(s) (MST) to solve these problems is proposed and, then, it is applied to three different types of environment for illustration. Computer simulation results show that our approach can obtain a satisfactory solution in moderate computational time.

I. Introduction

Flexible manufacturing systems (FMS's) become more and more popular nowadays in the field of manufacturing since they can perform different workpieces automatically in more than one sequence and, hence, possess considerable flexibility. In other words, no human intervention will be needed in determining system operations and in handling contingent incidents in such an automatic manufacturing environment. Design of an FMS will involve several levels of consideration and each level will endeavor to solve various problems. One of those levels is planning and scheduling where problems regarding how to perform *classification* and *selection* are intrinsic ones. In this paper, a efficient approach for solving these problems is proposed and is, later, applied to three different types of environment for illustration.

The first environment contains problems of so called group technology (GT) (see in [1]-[3]. Its philosophy is to perform necessary operations of processing a part by a group (cell) of machines which are physically close-by as much as possible in order to minimize the total handling time. Before such a technology can be applied, all the parts to be produced in a machine shop with lots of machines have to be sorted out into several families. Each family of parts is formed by judging the degree of similarity in size, geometry, method of manufacture, etc. Above all, all the operations involved in each family should be related to only a particular group of machines. Obviously, these are also known as classification problems.

The second one is tied with problems of the so called process plan selection [5][6]. Due to the very flexibility of an FMS, for each part there exists a number of different process plans and each plan specifies different requirements for tools, auxiliary devices, as well as operations (for example, machining operations) to be performed altogether at a given cost. Moreover, since there usually will be several parts to be manufactured at the same time, there may exist a conflict in using a single machine for manufacturing more than one part due to the choice of some possible combination of so many relevant process plans. Therefore, a mission for the production planner is to select one out of all possible combinations of all process plans so that the minimum number of fixtures, grippers, fedders, and tools will be required. In another words, the corresponding total cost will be the minimum over all possible choices.

Very often in a realistic situation, not only a subset of a set of process plans has to be chosen but also the sequence among process plans in that chosen subset has to be determined as well. This constitutes the third environment to be considered here where the class of problem, originated from that mentioned above, is terms as process plan sequencing problems. Of course, the sequence to be selected here will correspond to the minimum total cost as has been requested previously.

All these problems mentioned so far are NP-complete and, hence, the complexity of solving these problems is considerably high if its so called "optimal" solution is attempted. Thus, introduction of some good heuristics which can reduce complexity will be preferable in solving these problems in real practice. The efficient approach proposed here is based on the concept of minimum spanning tree [4] mixed with some relevant heuristics. For illustration of the performance, some computer simulation examples are provided with satisfactory results.

II. Problem Formulations

A. Group Technology Problem

Group technology is an approach that seeks to identify those attributes of a population that permit its members to be collected into groups, sometimes called families, so as to take advantages of their similarities in manufacturing and design. In a manufacturing system part similarities are of two types: design attributes (such as geometric shape and size) and manufacturing attributes (the sequence of processing steps required to make the part). Suppose there are k parts, $p_1 \cdots p_k$, which need to be manufactured and each part has m attributes. Then a k×m matrix PI can be built to indicate the parts' information. Every entry PI_{ai} is a number or a character, which represents the value of ith attribute of part p_a. For illustration, table 1 shows the information about ten parts each of which has twelves attributes.

Since each attribute has different effect to identifying a part, the Hamming distance has been modified by introducing the weight coefficient w_i for each attribute i. Recall that the weighted Hamming distance between any two parts, say, p_a and p_b is defined as:

$$w_1{*}d_1 + w_2{*}d_2 + \cdots + w_m{*}d_m$$

attr. \ part	1	2	3	4	5	6	7	8	9	10
attr1	0	1	1	0	1	1	0	0	0	1
attr2	3.6	2.8	3.3	3.6	3.4	2.6	4.2	3.0	4.0	2.7
attr3	3	3	4	4	5	3	2	3	3	4
attr4	B	A	A	A	B	A	A	A	B	A
attr5	0	1	1	1	1	0	0	1	1	1
attr6	62	59	58	61	60	58	62	60	59	62
attr7	R	B	G	G	G	B	R	R	R	G
attr8	-1	0	1	-1	1	1	1	0	1	-1
attr9	10	8	8	10	10	10	8	10	8	8
attr10	Y	N	N	N	N	N	Y	Y	Y	Y
attr11	2.4	3.1	2.8	2.7	2.8	2.9	2.2	3.0	2.9	2.7
attr12	A	C	C	C	D	B	B	A	B	C

Table1 Example1: Twelve Attributes for Ten Parts.

where d_i is the difference of the ith attribute between the two parts. Based on the part information matrix PI, we can first calculate the Hamming distance between every pair of parts, then a weighted Hamming distance matrix WH which indicates the degree of difference among every parts can be created accordingly. Thus, based on the part information matrix, we define the weighted Hamming distance matrix WH of which each entry, say, (a,b) denotes the weighted Hamming distance between part p_a and p_b. Accordingly, the matrix WH registers the degree of dissimilarity among all parts. For example, table 2 is the weighted Hamming distance matrix associated with the part information matrix shown in table 1. In this special case, if the ith attribute is a numerical value, then $d_i = |PI_{ai} - PI_{bi}|$, else $d_i = 0$ when $PI_{ai} = PI_{bi}$ and $d_i = 1$ otherwise.

By these definitions, a group technology problem can be stated as: Given a kxk weighted Hamming distance matrix WH, group those k parts into f families where f is a given number.

B. Process Plan Selection Problem

Most of the planning methods for automated manufacturing systems are based on the assumption that for each part there is only one process plan (defined as a sequence of operations) available. But in practical situation (e.g., in an FMS) one can generate a set of different process plans, whose attributes as well as costs may vary from one to the others. The former are defined as the auxiliary devices (such as fixtures, grippers, feeders, and tools) that each plan specifies whereas the latter is the manufacturing cost induced by that particular plan. By this observation, every process plan can be identified with its attributes (i.e., required auxiliary devices) and the associated cost. Suppose there are n auxiliary devices in a manufacturing system and at certain time there are p process plans to manufacture k parts, then the following incidence row vector is defined for each process plan P_i, $1 \leq i \leq p$:

$$x_i = [x_{1i}, x_{2i}, \cdots, x_{ni}]$$

where

$$x_{ti} = \begin{cases} 1 & \text{if the auxiliary device t is used in } P_i \\ 0 & \text{otherwises} \end{cases}$$

and a row vector C is used to denote costs of all the process plans, i.e.,

$$C = [C_1, C_2, \cdots, C_p]$$

where C_i is the cost for process plan P_i. An example for demonstration is shown in figure 1.

Like the first class of problems, after figuring out the incidence vector for each process plan, we can then construct a pxp weighted Hamming distance matrix D measuring dissimilarity among the plans. Each entry d_{ij} denotes the weighted Hamming distance between process plan P_i and P_j. Now the process plan selection problem can be defined as follows: Choose from the set U={1, ..., p} a subset $S = \{j_1, \ldots, j_k\}$ which contains one representative process plan j_i for each part i, $1 \leq j_i \leq p$, i=1, ..., k, such that the following is minimized:

$$\sum_{(j_{i1}, j_{i2}) \in A^S} d_{j_{i1}, j_{i2}} + \sum_{j_i \in S} C_{j_i} \qquad (1)$$

where $A^S = \{(j_1, j_2), (j_1, j_3), \ldots, (j_{k-1}, j_k)\}$. In words, to manufacture k parts we choose from the set of p process plans a subset of k process plans (counting the multiplicity) such that the total costs of these chosen plans plus a measure dissimilarity over those plans is minimized.

C. Process Plan Sequencing Problem

The class of problems considered here is originated from that given in subsection B but also takes into account the importance of the processing order of the manufacturing plans selected. This additional consideration will become imperative when the setup efforts caused by switching from one process plan to another are comparable to the total costs of the relevant process plans.

By use of the previous notation, we now formulate the problems as follows: Corresponding to the subset $S = \{j_1, \ldots, j_k\}$, we define the set of all possible permutations of the ordered kth tuple $(j_1, \ldots, j_k)$ as $\Pi(S)$ of which each element is denoted as $\pi = (i_1, \ldots, i_k) \in \Pi(S)$. Then, the objective is to minimize

$$\sum_{(j_{il}, j_{im}) \in B(\pi)} d_{j_{il}, j_{im}} + \sum_{j_i \in S} C_{j_i} \qquad (2)$$

where $B(\pi) = \{(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k)\}$ over all possible $S \subset U$ and all possible $\Pi(S)$.

III. Solving Problems of Classification and Selection Using Modified MST

In section 2, we have formulated the three classes of problems to be studied here. Later in this section we will show that by simply introducing some suitable modifications to the MST according to the problem nature, all these problems can be solved efficiently. It is noteworthy, however, that certain heuristics will have to be introduced when concepts

$$WH = \begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 0 & 13.5 & 14.7 & 7.3 & 11.6 & 12.5 & 7.8 & 6.2 & 10.5 & 9.2 \\ 2 & 13.5 & 0 & 4.8 & 9.4 & 9.9 & 6.5 & 12.3 & 7.3 & 7.4 & 7.5 \\ 3 & 14.7 & 4.8 & 0 & 8.4 & 7.1 & 6.8 & 12.5 & 10.5 & 7.8 & 7.7 \\ 4 & 7.3 & 9.4 & 8.4 & 0 & 7.3 & 11.2 & 12.1 & 6.9 & 11.6 & 5.9 \\ 5 & 11.6 & 9.9 & 7.1 & 7.3 & 0 & 8.7 & 14.4 & 8.6 & 9.7 & 10.8 \\ 6 & 12.5 & 6.5 & 6.8 & 11.2 & 8.7 & 0 & 12.3 & 8.5 & 9.4 & 13.3 \\ 7 & 7.8 & 12.3 & 12.5 & 12.1 & 14.4 & 12.3 & 0 & 10.0 & 6.9 & 10.0 \\ 8 & 6.2 & 7.3 & 10.5 & 6.9 & 8.6 & 8.5 & 10.0 & 0 & 7.1 & 9.6 \\ 9 & 10.5 & 7.4 & 7.8 & 11.6 & 9.7 & 9.4 & 6.9 & 7.1 & 0 & 11.5 \\ 10 & 9.2 & 7.5 & 7.7 & 5.9 & 10.8 & 13.3 & 10.0 & 9.6 & 11.5 & 0 \end{array}$$

Table2 Weighted Hamming Distance Matrix for Example 1.

$$
\begin{array}{ll|cccccccccc|}
 & & \overbrace{\quad\quad}^{\text{part 1}} & & \overbrace{\quad\quad\quad}^{\text{part 2}} & & & \overbrace{\quad\quad}^{\text{part 3}} & & \overbrace{\quad\quad\quad}^{\text{part 4}} & & \\
\text{tools} & t_1 & 1 & & 1 & 1 & & 1 & 1 & 1 & & 1 \\
 & t_2 & & 1 & & & 1 & & & 1 & & 1 \\
 & t_3 & 1 & 1 & 1 & 1 & 1 & 1 & & & 1 & \\
 & t_4 & 1 & & & & 1 & & 1 & 1 & 1 & 1 \\
 & t_5 & & 1 & & & & 1 & & & & 1 \\
\text{fixtures} & f_1 & 1 & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & f_2 & & 1 & & & 1 & & & 1 & & \\
 & f_3 & & 1 & 1 & & & 1 & & 1 & & 1 \\
 & & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10}
\end{array}
$$

$C = [5.8, 9.4, 11.6, 5.7, 3.4, 4.3, 5.1, 6.4, 5.2, 5.3]$

Figure 1. Example2: Ten Process Plans associated with Four Parts.

of MST are to be used to solve these NP-complete problems in general. Hence, the optimality of solutions will be traded with the efficiency of reaching good solutions.

A. Group Technology Problem

Recall that this class of problems are characterized mainly by the weight Hamming distance matrix WH. The objective is to group k parts in total into f families. Then, it is quite straightforward to build a graph G=(V,E) according to the matrix WH. Specifically, V corresponds to the set of parts and each element of E, say, (v,w) denotes the weighted edge between the part p_v and p_w. Now we are ready to propose an approach of solving the problems in the following:

Algorithm A

```
Step1:  According to the weighted Hamming distance matrix WH, build
        a corresponding undirected complete graph G=(V,E).
Step2:  Create a minimum spanning tree S=(V,T) of G.
Step3:  For index = 1 to f-1 do Step3.1 and Step3.2:
        Step3.1:  Choose an edge $(v,w)$ in T of the largest weight;
        Step3.2:  Delete $(v,w)$ from T.
Step4:  Now the graph S=(V,T) is an unconnected graph with f components.
```

After a minimum spanning tree is constructed in Step2, (f-1) edges of the tree with the larger weights are deleted so that S becomes an unconnected graph with f components. Each component thus corresponds to a family of parts. For example, there are totally ten parts which are shown in table 1 and table 2, and they are to be grouped into three families. Figure 3 shows the partial and the final result of our approach where three families are found as: $\{p_1,p_4,p_8,p_{10}\}$, $\{p_2,p_3,p_5,p_6\}$, and $\{p_7,p_9\}$.

Remark: To balance the load of these families the above algorithm should be modified to meet this constraint. At first an integer number *lp* indicating the lowest number of parts in every family must be given (or be calculated by some rule), then Step3 can be modified as follows (the

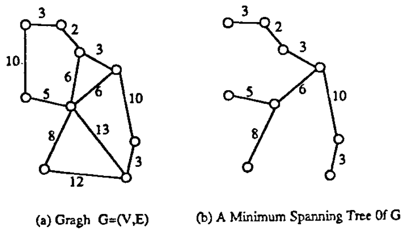


Figure 2. An Example of Minimum Spanning Tree.

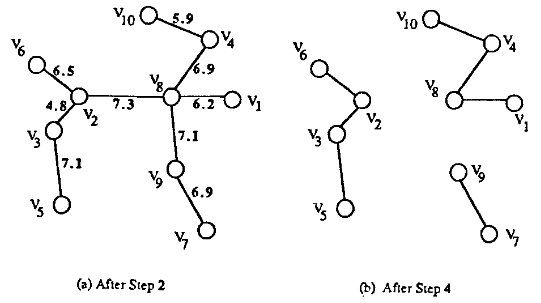


Figure 3. An Example of Grouping Result.

other steps remain unchanged): When the next largest edge is chosen, we must check after deleting this edge whether there exists a subtree whose number of node is less than *lp*, If it is, then this edge must be kept, and is deleted it otherwise. As before, after deleting f − 1 edges Step3 is then terminated.

B. Process Plan Selection Problem

The problems to be considered here are not only characterized by a weighted Hamming distance matrix D (as before) but also a cost vector C of all relevant process plans and a variable k which indicates the number of total parts. Likewise, an undirected graph G=(V,E) is constructed according to the matrix D and the vector C as well, where V now denotes the set of process plans whereas the weight of each edge in E, say, (v,w) is defined as

$$
\mid (v,w) \mid \; = C_v + C_w + \frac{C_2^k}{(k-1)} d_{vw} = C_v + C_w + \frac{k}{2} d_{vw} \qquad (3)
$$

where C_2^k denotes the number of 2-combinations of a k-set. The choice of the weight function is due to the fact that the total weight of any solution tree (with (k-1) tree branches) out of the graph G will generally be more close to the value given by the objective function in (1) than when d_{vw} instead of $\frac{C_2^k}{k-1} d_{vw}$ is used in the calculation (3). Based on the graph, a modified MST is constructed and gives a desired result. The detailed algorithm is shown below:

Algorithm B-1:

```
Step1:  According to the pxp weighted Hamming distance matrix D,
        the number of parts to be processed k, and the cost vector C, do
        Step1.1 and Step1.2 to build an undirected complete graph
        G=(V,E).
        Step1.1:  For each process plan $P_i$ builds a vertex $V_i$ in V so
                  that the number of vertices in V is p.
        Step1.2:  The weight of the edge (v,w) is given by (3).
Step2:  Let F be a set called forest, which is a set of trees. Initially, F
        is set to be empty.
Step3:  Do Step3.1 through Step3.3 repeatedly until F becomes a tree
        with k vertices.
        Step3.1:  Choose (v,w) an edge in E of lowest cost.
        Step3.2:  Delete (v,w) from E.
        Step3.3:  Test whether the edge (v,w) can be legally added
                  into the forest F and then add it if can.
Step4:  Now the forest F contains only one single tree. This implies
```

D=

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	2	1	3	3	1	4	1	4
2		∞	5	6	4	4	8	5	6	5
3			∞	∞	∞	1	3	4	3	4
4				∞	∞	2	2	5	1	5
5					∞	6	4	3	2	5
6						∞	∞	5	4	3
7							∞	3	2	3
8								∞	∞	∞
9									∞	∞
10										∞

Table 3. Weighted Hamming Distance Matrix for Example2.

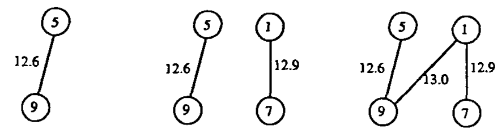
that the set of process plans represented by the corresponding vertices in that tree is a valid selection for processing the total k parts.

Note that the test made in Step3.3 is to see whether the addition of a candidate edge (v,w) into F will result in a cycle or will lead to the situation where more than one process plan are selected for the same part. To illustrate the algorithm, it is applied to example 2 shown in figure 1 with the corresponding graph shown in figure 4. It can be easily seen that the edge of the smallest weight is (v_5, v_9) and is, hence, added into F. Since P_4 and P_5 are both process plans of manufacturing part 2, the edge of the second smallest weight (v_4, v_9) can not be added into F. Continue this process and, then, we will have edges (v_1, v_7) and (v_7, v_9) be added into F successively. Now F becomes a spanning tree with four vertices (three branches) so that the algorithm terminates with the final solution to the selection, namely, $\{P_1,P_5,P_7,P_9\}$, in contrast with the optimal solution $\{P_1,P_4,P_7,P_9\}$. The total cost for the former is 32.5 whereas for the latter is 29.8. Apparently, the ratio of the difference to the optimal cost is about 9.0%.

In algorithm B-1 there exists only one forest in F at any time. In order to find a better solution we may change the algorithm to the one that F may contains more than one forest. Thus we have the following refined algorithm.

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	21.4	13.5	15.2	16.1	12.9	20.2	13.0	19.1
2		∞	31.0	27.1	20.8	21.7	26.9	22.2	23.0	24.7
3			∞	∞	∞	17.9	22.7	26.0	22.8	24.9
4				∞	∞	14.0	14.8	22.1	12.9	21.0
5					∞	19.7	16.5	15.8	12.6	18.7
6						∞	∞	20.7	17.5	15.6
7							∞	17.5	14.5	16.4
8								∞	∞	∞
9									∞	∞
10										∞

Table 4. The Weight of Every Edge E_{ij} in the Graph G=(V,E) Built by Example2.



(a) After Adding the Edge of the Smallest Weight.
(b) After Adding the Edge of the Third Smallest Weight.
(c) After Adding the Edge of the Forth Smallest Weight.

Figure 4. Running Algorithm B-1 and C-1 for Example2.

Algorithm B-2:

Step1: According to the p*p weighted Hamming distance matrix D, the number of parts wanted to be processed k, and the cost vector C, do Step1.1 and Step1.2 to build an undirected complete graph G=(V,E).

Step1.1: For each process plan P_i builds a vertex V_i in V so that the number of vertices in V is p.

Step1.2: The weight of the edge (v,w) is given in (3).

Step2: Let F be a set of forests. Initially there is only one empty set in F, i.e., an empty forest.

Step3: Define a variable fn as the number of forests in F. Initially fn is set to be one.

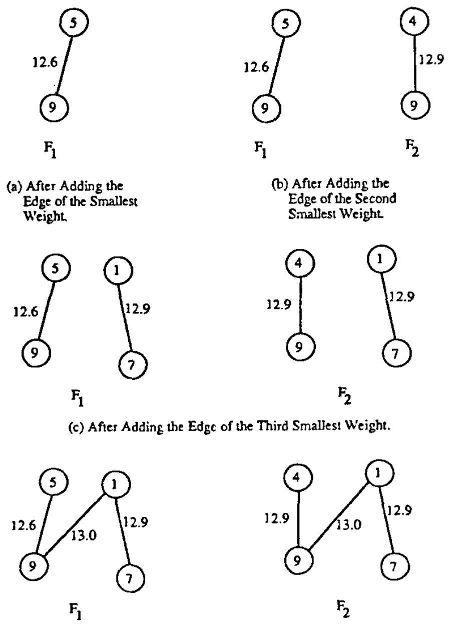
Step4: Do Step4.1 through Step4.4 repeatedly until one of the forests in F is a tree with k vertices.

Step4.1: Choose (v,w) an edge in E of lowest cost.

Step4.2: Delete (v,w) from E.

Step4.3: For i=1 to fn, test whether the edge (v,w) can be legally added into the forest F_i and then add it if can.

Step4.4: If the edge (v,w) can't be legally added in any forest F_i then do Step4.4.1 and Step4.4.2.



(a) After Adding the Edge of the Smallest Weight.
(b) After Adding the Edge of the Second Smallest Weight.
(c) After Adding the Edge of the Third Smallest Weight.
(d) After Adding the Edge of the Forth Smallest Weight.

Figure 5. Running Algorithm B-2 and C-2 for Example2.

```
        Step4.4.1:   $fn \leftarrow fn + 1$.
        Step4.4.2:   Clear the forest $F_{fn}$ to be empty, and, then, add the edge (v,w) into this new forest.
Step5:  Now there is at least one forest $F_i$ in F become a tree with k vertices. If there exists more than one such trees in F then find a tree $F_{min}$ with smallest cost according to the cost function (1). Then the set of process plans represented by the corresponding vertices in $F_{min}$ is a valid selection for processing the total k parts.
```

The major difference between algorithm B-2 and B-1 is that F in algorithm B-2 is a set of forests instead of a set of trees, and a new forest F_{fn+1} with a single edge (v,w) must be constructed if the edge (v,w) can't be legally added into any forest F_i, $1 \leq i \leq fn$. For illustration figure 5 shows the forests built by algorithm B-2 for example 2. Figure 5(b) shows that the second smallest edge (v_4, v_9) can't be legally added into the forest F_1 and, hence, a new forest F_2 is then constructed. In figure 5(c) the third smallest edge (v_1, v_7) can be legally added into both F_1 and F_2 so that it is incorporated into both of them. At last, there are two spanning trees, F_1 and F_2, with four vertices as shown in figure 5(d). After the total cost of these two trees is calculated, the set of process plans $\{P_1, P_4, P_7, P_9\}$ is the final solution which is also the optimal solution.

C. Process Plan Sequencing Problem

The necessary information for this problem is the same as the above one, i.e., a weighted Hamming distance matrix D, a variable k indicating the number of parts wanted to be processed, and a cost vector C. The algorithms for solving this problem are similar to that for solving the previous problem except that the testing procedure is different and the weight in every edge when building the corresponding graph G is also different. Specifically, the weight for the edge (v,w) is defined as the sum of C_v, C_w, and $2 * d_{vw}$ in view of the fact that the total weight of edges in a sequence is twice of that given by (2) according to this weight definition. Furthermore, when the edge (v,w) is chosen, we must test after it is added whether the forest can form a valid sequence instead of whether it can form a valid spanning tree. Two algorithms for solving this class of problems are shown as follows.

Algorithm C-1:

```
Step1:  According to the pxp weighted Hamming distance matrix D, the number of parts to be processed k, and the cost vector C, do Step1.1 and Step1.2 to build an undirected complete graph G=(V,E).
        Step1.1:   For each process plan $P_i$ builds a vertex $V_i$ in V so that the number of vertices in V is p.
        Step1.2:   The weight of edge (v,w) is the sum of $C_v$, $C_w$, and $2 * d_{vw}$.
Step2:  Let F be a set of trees, that is, a forest. Initially, F is an empty set with no element.
Step3:  Do Step3.1 through Step3.3 repeatedly until F becomes a sequence with k vertices.
        Step3.1:   Choose (v,w) an edge in E of lowest cost.
        Step3.2:   Delete (v,w) from E.
        Step3.3:   Test whether the edge (v,w) can be legally added into forest F and then add it if can.
Step4:  Now the forest F contains only one sequence S. This implies that the order of process plans represented by the corresponding vertices in S is a valid sequence for processing the total k parts.
```

Algorithm C-2:

```
Step1:  According to the pxp weighted Hamming distance matrix D, the number of parts to be processed k, and the cost vector C, do Step1.1 and Step1.2 to build an undirected complete graph G=(V,E).
        Step1.1:   For each process plan $P_i$ builds a vertex $V_i$ in V so that the number of vertices in V is p.
        Step1.2:   The weight of edge (v,w) is the sum of $C_v$, $C_w$, and $2 * d_{vw}$.
Step2:  Let F be a set of forests. Initially, there is only one empty set in F which indicates that there is an empty forest in F at first.
Step3:  Define a variable fn as the number of forests in F. Initially fn is set to one.
Step4:  Do Step4.1 through Step4.4 repeatedly until there is a forest $F_i$ in F being a valid sequence with k vertices.
        Step4.1:   Choose (v,w), an edge in E of lowest weight.
```

size (p,k)	computer run time			ratio of finding optimal solution		DRATIO#		improvement by B-2	
	B-1	B-2	OPT*	B-1	B-2	B-1	B-2	extra time needed	improvement
(20,4)	0.0163	0.0203	0.0286	10 %	26 %	13.24 %	11.02 %	24.49 %	2.702 %
(20,5)	0.0197	0.0233	0.0823	8 %	22 %	14.80 %	11.41 %	18.64 %	2.994 %
(25,5)	0.0333	0.0393	0.2863	10 %	12 %	14.92 %	11.29 %	18.01 %	1.882 %
(30,5)	0.0513	0.0567	0.7626	10 %	16 %	14.71 %	13.70 %	10.39 %	0.854 %
(30,6)	0.0520	0.0783	0.9081	6 %	10 %	16.69 %	11.05 %	50.62 %	5.372 %
(40,8)	0.1123	0.1697	40.032	0 %	2 %	16.85 %	14.11 %	51.08 %	2.314 %
(40,10)	0.1233	0.3887	56.152	0 %	0 %	13.81 %	9.82 %	215.1 %	3.528 %
(50,10)	0.1967	0.4231	803.43	0 %	0 %	16.18 %	13.14 %	174.4 %	2.582 %
(50,12)	0.2217	0.7082	1261.6	0 %	0 %	14.53 %	10.11 %	219.3 %	3.695 %

run time : CPU run time (sec) on VAX 8530

OPT*: exhaustive method for finding optimal solution.

DRATIO#: difference ratio between approximate solution and optimal solution.

Table 5. Performance of Algorithm B-1 and B-2.
(average value for 50 instances each problem solved)

```
    Step4.2:   Delete (v,w) from E.
    Step4.3:   For i=1 to fn, test whether the edge (v,w) can be legally added in forest $F_i$ and then add it if can.
    Step4.4:   If the edge (v,w) can't be legally added in any forest $F_i$ then do Step4.4.1 and Step4.4.2.
        Step4.4.1:   fn $\leftarrow$ fn + 1.
        Step4.4.2:   Clear the forest $F_{fn}$ to be empty then add (v,w) in this new forest.
Step5:   Now there is at least one forest $F_i$ in F become a valid sequence with k vertices. If there exists more than one such sequences in F then find a tree $F_{min}$ with the smallest cost according to the cost function (2). Then the sequence of process plans represented by the corresponding vertices in $F_{min}$ is a valid one for processing the total k parts.
```

For example, after the edge (v_1, v_4) is added into the forest shown in figure 6(a) the forest in figure 6(b) is not a valid sequence although it is a valid tree. Application of these two algorithms to example 2 are shown in figure 4 and figure 5 respectively. Then forests built by these algorithms are found to be the same as the ones built by algorithm B-1 and B-2 because the weights $C_2^4/(4-1)$ (used in algorithm B-1 and B-2) are equal to 2 (use in algorithms C-1 and C-2). In figure 4 the process sequence obtained is $P_5 \rightarrow P_9 \rightarrow P_1 \rightarrow P_7$ (or $P_7 \rightarrow P_1 \rightarrow P_9 \rightarrow P_5$) which is the optimal solution. In figure 5(d) two valid sequences are obtained. After their respective costs are calculated, the final solution is the same as the above one.

IV. Computer Simulation Examples

In this section, we implement some simulation programs of the algorithm B-1, B-2, C-1, and C-2 on VAX 8530 to solve the relevant problems. Comparison between results of our approachs and the optimal solutions is performed and shown. In order to analyze the performance of these algorithms, random problems of various sizes (p,k) were generated. Here p is the number of process plans and k is the total number of parts. The cost C_i, for each $1 \leq i \leq p$, were uniform in interval (0,15) and the Hamming distances were random integers in [1,10]. Table 5 and 6 summarize the results obtained.

The first column of table 5 shows the computer run time of three algorithms: B-1, B-2, and a exhaustive search method to find the optimal solution (say, OPT method). Since the process plan selection is NP-complete, the run time of OPT method grows greatly as the problem size increases. The result shows that algorithm B-1 and B-2 are both very efficient because of the little run time. The second column demonstrates the percent of finding the optimal solution of algorithm B-1 and B-2. It is shown that it will become more and more difficult to find the optimal solution for these methods as the problem size increases. The third column lists the difference ratio between the optimal solution and that our approach found. The result shows that the value obtained from algorithm B-2 is about ten to fourteen percent off the value of best solution, and the solution obtained from algorithm B-1 is a little worse than that from algorithm B-1. The last column reveals that the improvement of solution quality is about 2 to 6 percent of algorithm B-2, but the computer time needed is much longer than that of algorithm B-1 as the problem size increases. Finally it seems to make the application of algorithm B-2 worthwhile.

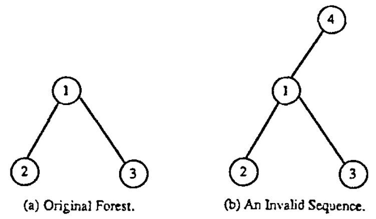


Figure 6. Example of an Invalid Sequence.

size (p,k)	computer run time			ratio of finding optimal solution		DRATIO#	
	C-1	C-2	OPT*	C-1	C-2	C-1	C-2
(15,5)	0.0125	0.0208	1.3375	30 %	30 %	9.63 %	11.12 %
(20,4)	0.0183	0.0242	0.5075	20 %	20 %	7.61 %	11.52 %
(20,5)	0.0201	0.0267	5.7421	0 %	10 %	11.18 %	17.60 %
(25,5)	0.0375	0.0442	21.143	10 %	10 %	14.26 %	13.38 %
(30,5)	0.0528	0.0639	52.556	10 %	10 %	10.05 %	18.41 %
(30,6)	0.0667	0.1010	350.34	0 %	0 %	6.97 %	13.81 %

run time : CPU run time (sec) on VAX 8530.

OPT* : exhaustive method for finding optimal solution.

DRATIO# : difference ratio between approximate solution and optimal solution.

Table 6. Performance of Algorithm C-1 and C-2.
(average value for 10 instances each problem solved)

The similar results are shown in table 6 for algorithm C-1 and C-2. To sum up, we can see that algorithm C-1 is a very efficient and satisfactory method when applied to the process plan sequencing problem.

V. Conclusion

An efficient method of solving problems of classification and selection using minimum spanning tree in an FMS is proposed in this paper. Computer simulation examples are then provided which shows a satisfactory result. Especially, the total computational time spent is also economical. The application of this method to these classes of problems will be quite promising. Ongoing research will be on considering the very flexibility nature of an FMS and then extending this method to solve more dynamic and general problems in an FMS.

VI. Reference

[1] I. Ham, K. Hitomi, and T. Yoshida, *Group Technology*, Kluwer Nijhoff publishers, Hingham, Mass., 1985.

[2] C. C. Gallagher and W. A. Knight, *Group Technology*, Butterworth & Company (Publishers) Ltd., London, 1973.

[3] A. Kusiak, A. Vannelli, and K. R. Kumar, "Grouping problem in scheduling flexible manufacturing systems," *Robotica*, vol. 3, pp. 245-252, 1985.

[4] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *in Proc. Amer. Math.*, Soc. 7:1, 48-50.

[5] T. C. Chang and R. Wysk, *An Introduction to Automated Process Planning Systems*, Englewood Cliffs, HJ: Prentic-Hall, 1985.

[6] A. Kusiak and G. Finke, "Selection of process plans in automated manufacturing systems," *IEEE J. of R. and A.*, vol. 4, no. 4, Aug. pp. 397-402, 1988.