Planning and Scheduling in a Flexible Manufacturing System Using a Dynamic Routing Method for Automated Guided Vehicles

Pei-Sen Liu and Li-Chen Fu

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

ABSTRACT

After a flexible manufacturing system (FMS) is built and configured, the two main problems left to be solved are planning and scheduling. In this paper, a new approach that can dynamically solve the planning and scheduling problem in an FMS is presented. This problem is formulated as the determination of an optimal routing assignment of p automated guided vehicles (AGV's) among m workstations in order to accomplish N tasks in an FMS. First we introduce a useful task representation, called "workgraph", to facilitate our later computation, and then we use the A^* search algorithm, minimax criterion, and some heuristic rules to solve this routing assignment problem dynamically. Our approach obtains a near-optimal solution in moderate computation time, and, in addition, solves some dynamic situations so as to make the FMS more flexible.

I. Introduction

An FMS is a large complex system consisting of many interconnected components of hardware and software, such as, pallets, AGV's, fixtures, and tool capacity [1] [2]. In an FMS, parts are automatically transported via AGV's from one workstation to another for processing under computer control. Several aspects of current FMS planning and decision control system have been discussed in [2]-[6]. Due to the very flexibility of these planning and decision control system, the productivity of an FMS is usually much higher than that of a classical manufacturing system, in particular, when various parts of many kinds and of small volumes are produced.

Several approaches to planning, decision control, and scheduling of an FMS have been suggested in [7]-[11]. However, the proposed works can hardly solve the problem of planning and scheduling dynamically, nor they are suitable for solving the routing problem of those indispensable AGV's in an FMS. Chen et al. in [15] formulate the task assignment problem as a routing problem of autonomous vehicles and then solve it. But the representation for tasks used there is considerably simplified and their method fails to be a dynamic scheme. In this paper, a new approach aimed at improving these shortings is proposed.

This paper formulates the problem of dynamic planning and scheduling as the one of determining the optimal routing assignment of p AGV's among m workstations to accomplish N tasks in an FMS. The approach used here is a dynamic one which solves the problems of planning and scheduling as a whole. Especially it can handle some unexpected situations, e.g., when some workstations break down, as well as can respond to some changes, such as, of production target. In addition, since the number of total tasks needed to be accomplished is usually very large (in hundreds, thousands, or more), such a method can be regarded as an implementation of an integrated version of medium-term and short-term decision supporting software as described in [2].

The layout of this paper is as follows: in section II, we formulate the problem and describe the optimality criterion for the solution; in section III, we introduce a useful representation of each task ready to be handled, and called it "workgraph"; section IV explains how we use A* search algorithm and minimax strategy for solving this routing problem; in section V, we propose an approach of dynamic routing assignment in an FMS.

II. Problem Formulation and Optimality Criterion

The main problem left to be solved after an FMS is built and configured is how to run the manufactory efficiently to get the maximum productivity. This problem is usually divided into two parts: planning, or process routing, which is the selection of a sequence of operations; and scheduling, which is the assignment of time and resources. If we take these two problems into consideration simultaneously and find a method to solve them dynamically, then the productivity may be improved greatly and the manufacturing system may become more flexible under many dynamic situations. Hence we define our problem as involving the determination of an optimal routing assignment of p AGV's among m workstations to accomplish N tasks in order to minimize the total completion time.

Among N tasks, supposed there are k kinds of tasks and the number of the ith kind is N_i, $1 \le i \le k$, so that $N = N_1 + N_2 + \ldots + N_k$. To accomplish a task is to let an AGV carry necessary materials for this task (parts or subparts) and to assign the route among workstations so that every subtask of the task can be successfully processed by the selected workstations. Thus, completing N tasks is to assign routes for p AGV's among m workstations. Moreover, the total completion time of a task is the sum of total execution time (through workstations), total travelling time of an AGV (among workstations), and total waiting time (when two or more AGV's arrive at the same workstation). Now due to the fact that a task is accomplished through a sequence of processing by some selected workstations, performing the routing assignment is equivalent to

solving the problem of planning. Furthermore, because we use the completion time as the main factor to assigning the route for every AGV, the planning problem can also be treated as a scheduling problem. Consequently, we can define the problem of planning and scheduling together in an FMS as the routing assignment problem, which can also be treated as an integration of medium-term and short-term decision problem. Now various assumptions made about the AGV's, workstations, and tasks are discussed in the following.

Assumptions :

1) All AGV's are the same. Every AGV can carry all kinds of materials necessary for all kinds of tasks. This implies that the amount of processing time spent in any workstation to accomplish a particular kind of subtask is the same for all AGV's. But the starting positions of all AGV's are not necessarily the same, i.e., each AGV may start from any dispatching center.

2) All AGV's are travelling at constant speed but the speed of every AGV may be different. This facilitates the detection of collision among AGV's at any workstation. It also allows us to calculate the arrival time of every AGV at any workstation in order to find the waiting time needed for other AGV's when collisions are about to occur.

3) The positions of all workstations are fixed and predetermined. Thus the travelling time of any AGV from its starting position to any workstation, or from one workstation to another is predetermined.

4) All workstations along with all paths connecting them form a complete graph. This implies that there is exactly one (undirected) edge between any two workstations, and any workstation can be reached from any other workstation by AGV's.

5) There are no precedence constraints between any two tasks. And the task which can be handled here is the one that can be represented by a "workgraph" which will be clearly defined in the sequel. Denote the class of such tasks by H.

number of every part to be produced is predetermined. This information is useful for the heuristic estimation in our later dynamic planning and scheduling algorithm.

7) If an AGV has not finished the task assigned, it will not be involved in the execution of another task.

Remark: The tasks which can be represented by workgraphs form, in fact, a subset of the tasks that can be represented by general AND/OR graphs.

A. Representation of the System Model

A material handling system of an FMS can be represented by a triplet $S = (A, W, T)$, in which the arguments represent the set of *AGV's* , the set of *workstations*, and the set of *task* graphs respectively. Specifically, $A = \{ v_1, v_2, ..., v_p \}$, where v_i denotes the ith AGV, and all p AGV's are the same; $W = \{ W_1, ..., W_q, W_{q+1}, ..., W_m \}$, where W_i, $1 \le i \le q$, denotes the set of real workstations and W_j, $q+1 \le j \le m$, denotes the set of terminals for AGV's (loading and unloading places); and $T = \{ T_1, T_2, ..., T_k, N_1, N_2, ..., N_k \}$, where T_i denotes the AND/OR task graph representing the ith kind of tasks that belong to H and N_i the number of tasks of the same kind, for all $1 \le i \le k$, and $N = N_1 + ... + N_k$.

Remarks:

(1) Later in the sequel, we will not distinguish the real workstations from terminals, and simply call W_i the ith workstation, $1 \le i \le m$.

(2) We further denote $T_i = (V_i, E_i)$, the ith task graph, where V_i represent the set of vertices and E_i the set of directed links.

Because we assume the positions of all workstations are fixed and predetermined, the distance between every two workstations is known in advance. Furthermore, since the speed of all AGV's are constant, we can henceforth use a symmetric matrix to represent the travelling time for an AGV between any two workstations. We now let WTT^r be this matrix, where the entry $WTT^r_{i,j}$ denotes the travelling time spent by the AGV v_r when it travels from workstation i to workstation j, $1 \le i,j \le m$. Every task graph $T_i = (V_i, E_i)$ is an AND/OR graph with directed edges. The directed edge from vertex i to vertex j indicates that the subtask to be processed by workstation i must go before the subtask to be processed by workstation j. An OR branch indicates that only one of the successive subtasks associated with the branch needs to be accomplished whereas an AND branch indicates that they all have to be accomplished without any precedence and dependency relationships among them (so that the order to accomplish these subtasks is not important). We also note that in a real material handling system of an FMS, the number of vertices with zero indegree edge is always one and the number of vertices with zero outdegree edge is also one, representing the loading and unloading places respectively. For illustration, Table 1 and Figure 1 show a typical material handling system with two AGV's, seven workstations, and three different kinds of tasks. The number of every kind of tasks is 100, 50, 75 respectively, and the number in every vertex indicates the amount of execution time required in this workstation to accomplish the corresponding subtask.

Table 1. Workstation travelling time matrix.

	W1	W2	W3	W4	W5	W6	W7
W1	0	1	5	8	6	4	9
W2	1	0	4	3	5	6	4
W3	5	4	0	4	7	5	3
W4	8	3	4	0	8	7	2
W5	6	5	7	8	0	6	7
W6	4	6	5	7	6	0	8
W7	9	4	3	2	7	8	0

for AGV v_1

	W1	W2	W3	W4	W5	W6	W7
W1	0	1	5	7	6	3	8
W2	1	0	4	3	5	6	4
W3	5	4	0	4	6	5	3
W4	7	3	4	0	8	7	3
W5	6	5	6	8	0	5	6
W6	3	6	5	7	5	0	7
W7	8	4	3	3	6	7	0

for AGV v_2

B. Cost Function

After using an workgraph to represent all possible ways to accomplish a task, we must choose a path from the initial state to the final state to accomplish the task. It is equivalent to performing planning in an FMS, that is to find a sequence of operations to accomplish the task. The cost function used for searching the optimal planning is the sum of execution time through workstations, travelling time of an AGV among workstations, and waiting time of that AGV for avoiding collisions when reaching workstations. Now suppose that the AGV v_i is assigned to perform tasks $t^i_1, ..., t^i_{P_i}$ sequentially, then its travelling sequence is denoted $O^i = (O^i_1, O^i_2, ..., O^i_{n_i})$ where $O^i_j \in W$, $1 \le j \le n_i$, and n_i is the total number of workstations travelled. Note that a workstation may be

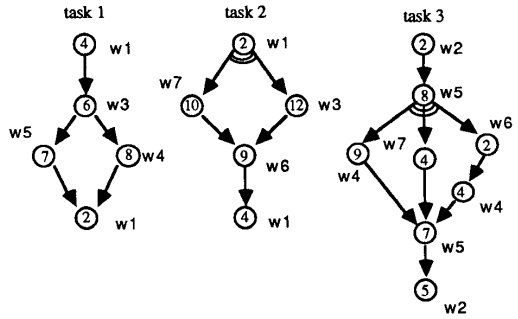

Figure 1. System graph for p=2, m=6, k=3, N1=100, N2=50, N3=75, $T_i = (V_i, E_i)$ is the ith kind of task graph for all i=1,2,3.

travelled through more than once, and the sequence of workstations to be followed by any AGV for travelling when accomplishing tasks. Given this notation, we are now ready to define travelling time, execution time, and waiting time throughout a task execution by an AGV, say, v_i in detail as follows.

Definition 1: Let TT^i_j denote the time spent by the AGV v_i when travelling from workstation O^i_j to workstation O^i_{j+1} out of the sequence O^i, $1 \le j \le (n_i - 1)$. Then the travelling time of v_i to accomplish the sequence of tasks $t^i_1, \ldots, t^i_{p_i}$ is defined as:

$$TT^i = \sum_{j=1}^{n_i - 1} TT^i_j$$

Definition 2: Let E^i_j denote the time required by workstation O^i_j to complete its processing, $1 \le j \le n_i$. Then task execution time corresponding to the travelling sequence O^i is defined as:

$$ET^i = \sum_{j=1}^{n_i} ET^i_j$$

Definition 3: If AGV v_i and AGV v_j arrive at a workstation at the same time or a workstation is doing some subtask for AGV v_j when AGV v_i arrives, then either of the two AGV's, v_i and v_j, has to wait or v_i has to wait till the workstation finishes the current subtask. Hence we let WT^i_k be the waiting time of the AGV v_i at the workstation O^i_k from the travelling sequence O^i so that the total waiting time of the AGV v_i to complete the travelling sequence O^i is defined as:

$$WT^i = \sum_{j=1}^{n_i} WT^i_j$$

As a result of the above definitions, the total completion time $T_i(S)$ spent by the AGV v_i corresponding to the routing assignments S for all AGV's is the sum of travelling time, task execution time, and waiting time as follows:

$$T_i(S) = TT^i + ET^i + WT^i$$

C. Optimality Criterion

The purpose of the routing assignment in this problem is to determine an optimal travelling sequence for p AGV's among m workstations in order to accomplish total N tasks. Let Ω denote all possible routing assignments for p AGV's to accomplish all tasks. Then our optimality criterion is to find an optimal routing assignment S such that:

$$S = \underset{S \in \Omega}{\operatorname{argmin}} \{ \max_{1 \le i \le p} (T_i(S)) \}$$

It is well known that this problem is a NP-complete problem. Also in real practice, the number of each kind of tasks is quite large so that the optimal routing assignment can hardly be found in a reasonable period of time. Moreover, the environment of a manufacturing system may be changing dynamically (e.g., some N_i may be increased, or a workstation suddenly breaks down). Therefore, it may be impractical to spent a lot of time to find an optimal assignment in much advance of real running. In the following, we will use a dynamic method to find a "near optimal" solution.

III. Workgraph

The first step of our approach is to find a good representation of tasks to be accomplished. In an FMS, there are usually many workstations installed, and in many cases there may be more than one workstation which can perform a particular subtask. As a result, there may be more than one way to accomplish a task. In [9] Fox and Kempf used a precedence diagram to represent the possible solutions. However, their representation is not general enough to represent all possible sequences. In fact, it can not represent the "OR" relation in an AND/OR graph which may occur often in an FMS. Homem in [11] used an AND/OR graph to represent a decomposable assembly task, and his representation turns out to be more general than the others. In this section, we will introduce "workgraph" which is also a useful representation. Although a workgraph is not as general as an AND/OR task graph, it can include "OR" relation in the graph. Hence, here we will contend ourselves by only considering a subclass of all decomposable tasks that can be represented by our workgraph. The following is the recursive definition of our workgraph:

Definition 4: A *workgraph* is a particular graph (V, E), where V is the set of vertices, and E represents the set of directed edges which connect all the vertices in V in a single chain. Every vertex has one of the following interpretations:

(a) A workstation: which indicates that a subtask can be accomplished by this workstation;

(b) Some other workgraphs with "OR" relationship: which implies that only exactly one of these workgraphs will be chosen.

(c) Some other workgraphs with "AND" relationship: which implies that all these workgraphs have to be chosen once (the order of choices is non-important).

Several advantages of using workgraphs can be described as follows. First, due to the hierarchical structure of workgraphs, a task can be more clearly represented and choices as to accomplishing the task can be more easily understood. Secondly, the heuristic estimation h(n) of every node in expanding a workgraph when using A^* search algorithm can be very easy and precise. This point will be explored more in detail in next section. Finally, for our later dynamic routing assignment algorithm, the idea of ordering all the vertices in a sequence is very useful, and, particularly, it will facilitate our dynamic routing assignment algorithm greatly. For illustration, Figure 2 shows the workgraphs corresponding to the task graphs of Figure 1.

The procedure of constructing a workgraph is given in [16].

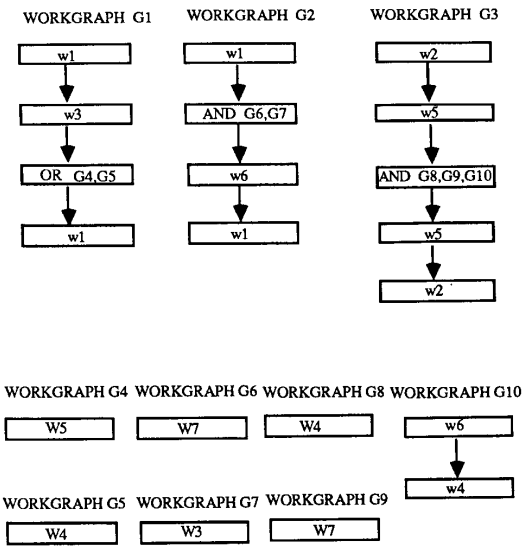

Figure 2. Workgraphs for task 1, 2, 3.

IV. A* Search Algorithm and Minimax Strategy

After using workgraphs to represent tasks to be accomplished, we can now apply our dynamic routing assignment algorithm to find a near optimal solution. Because the algorithm involves repeated use of A* algorithm (which will be described in detailed in next section), in this section we will briefly review the A* search algorithm and the minimax strategy, and exemplify them by solving the subproblem in our later dynamic planning and scheduling algorithm: given p tasks in total to be executed by p AGV's, what is the optimal routing assignment for all p AGV's among m workstations?

This subproblem can be formulated as a state space search problem, and the well known A* algorithm and minimax strategy can be used for solving this subproblem. In our subproblem here, a solution is an optimal routing assignment for p AGV's among m workstations to accomplish p tasks. Suppose that the tasks assigned to AGV v_1 ..., v_p are t_1, ..., t_p respectively, then let WG_1, ..., WG_p be their corresponding workgraphs. We now give the state space search method as follows.

1) *State Description* : Let an ordered triplet set $O = \{ (i,R^i,O^i) \mid 1 \le i \le p \}$ denote a current routing assignment for p AGV's, where O^i represents the current travelling sequence for the AGV v_i as mentioned before, and $R^i = \{ ((e^i_1,l^i_1), (e^i_2,l^i_2), ...\}$ represents the time history that records the entering time and the departing time at every workstation for this AGV v_i, i.e., each ordered pair (e^i_j,l^i_j) indicates that the time when the AGV v_i enters and leaves the workstation O^i_j is e^i_j and l^i_j respectively. Thus each triplet (i,R^i,O^i) represents that so far the AGV v_i is assigned to the route O^i and the current travelling time history is R^i.

2) *Initial State* : The initial state is $O = \{ (i,\phi,\phi) \mid 1 \le i \le p \}$ which means that no travelling sequence is assigned to any AGV.

3) *Operator* : In the A* search algorithm, the node to be expanded in every step is determined by the so called evaluation function f(n), where n denotes a node. The function f(n) is a cost estimate of the solution path in the state space which is constrained to pass the node n. Usually, it can be written as:

$$f(n) = g(n) + h(n)$$

where g(n) denotes the cost associated with the path from the starting node to the current node n, and h(n) denotes the heuristic cost from the node n to a goal node. In our case, within each node, we assign a cost estimate to each ordered triplet, namely, $f_i(n)$, i= 1, ..., p, and then we define

$$f(n) = \max_i f_i(n)$$

Now if the node, say, n is chosen to be expanded next, the way we expand it is to change an ordered triplet, say, (j,R^j,O^j) to (j,R'^j,O'^j), where f_j is the smallest among $f_i(n)$, i= 1, ..., p, and O^j is not complete in terms of its assigned task, by adding a workstation according to its workgraph to get O'^j and finding its corresponding R'^j. Consequently, the descendant node will be of the form

$$\{ (1,R'^1,O^1), ...,(j,R'^j,O'^j), ...,(p,R'^p,O^p) \}$$

where the original time travelling sequence R^i may be changed to R'^i, $i \neq j$ due to the change of O^j, but O^i, $i \neq j$ remain the same. Moreover, if there exist more than one choice to change (j,R^j,O^j) to (j,R'^j,O'^j), e.g., when the added workstation turns out to be the ORed workgraph, then more than one descendant nodes should all be found. In this way we can develop the partial routing assignments till all AGV's have accomplished their assigned tasks according to the corresponding workgraphs.

4) *Goal State* : Any state $O = \{ (i,R^i,O^i) \mid 1 \le i \le p \}$ is a goal state if and only if every O^i is a completed travelling sequence, i= 1, ..., p, i.e., every workgraph WG_i is traversed following the sequence O^i, i= 1, ..., p. Each R^i is then the final travelling time history for AGV v_i to accomplish its assigned task, and the total amount of time needed to accomplish all these tasks is $\max_i l^i_{k_i}$ where k_i is the ith number of workstations (counting the multiplicity) travelled by the AGV v_i.

Next we will describe how to estimate the heuristic cost $h_i(n)$ for the ith ordered triplet (i,R^i,O^i) in the node to be expanded. But before this, we will first introduce the heuristic cost associated with each vertex of a workgraph in the following procedure.

Heuristic Estimation Procedure :

Step 1. For every vertex V_i in the workgraph, do Step 1.1 and Step 1.2 to find the starting workstation set S_i and the final workstation set F_i.

Step 1.1. If the vertex consists of one workstation, then this workstation is the only element in the set S_i and the set F_i.

Step 1.2 If the vertex contains some workgraphs related by AND or OR, then the set S_i is the union of the starting workstation sets of these workgraphs, and the set F_i is the union of the final workstation sets of these workgraphs. Here, the sets S_i and F_i of the workgraph are actually obtained in a recursive way.

Step 2. For every vertex V_i in the workgraph, define the rough heuristic cost, denoted rh_i, as the minimum time needed to accomplish this vertex, which can be computed by performing the following steps.

Step 2.1. If the vertex contains only one workstation, then the rough heuristic cost rh_i is the execution time

spent in this workstation to accomplish the subtask.

Step 2.2. If the vertex contains some workgraphs related by OR, the the rough heuristic cost rh_i is equal to the smallest one of the rough heuristic costs of those workgraphs.

Step 2.3. If the vertex contains some workgraphs related by AND, then the rough heuristic cost rh_i is equal to the sum of all the rough heuristic costs of those workgraphs.

Step 3. If the number of vertices in this workgraph is n, then the heuristic cost hh_n for vertex n is zero.

Step 4. For i= n-1 down to 1, do the following steps to find the heuristic cost hh_i for vertex V_i, i= 1, ..., n-1.

Step 4.1 Choose two workstations W_k and W_l from F_i and from S_{i+1} respectively, such that the travelling time of, say, AGV v_r between W_k and W_l is the minimum and denote it tt_i.

Step 4.2. The heuristic cost h_i for vertex V_i is the sum of hh_{i+1}, tt_i, and rh_{i+1}. It means that the minimum cost from this vertex to the last vertex is hh_i.

Step 4.3. If the vertex contains some workgraphs related by OR or AND, then, when calculating the heuristic costs of these workgraphs, the sum of hh_{i+1}, tt_i, and rh_{i+1} should be added in order to get a more precise estimate.

Step 5. The heuristic cost of this workgraph is, hence, the heuristic cost of vertex one, i.e., hh_1.

After computing the heuristic cost of every vertex in the workgraph for all AGV's, we can then calculate every cost estimate $f_i(n)$ for every AGV v_i with ordered triplet (i,R^i,O^i) at run time. Recall that the definition of $f_i(n)$ is

$$f_i(n) = g_i(n) + h_i(n)$$

where $g_i(n)$ is the total cost for the AGV v_i to complete the travelling sequence O^i, i= 1, ..., p, and is calculated precisely according to the current routing information of all AGV's. If the AGV v_i is at the vertex V_{i_y} in its workgraph after completing this sequence O^i, then the heuristic function $h_i(n)$ is equal to hh_{i_y}, where hh_{i_y} is the heuristic cost of vertex V_{i_y}. Thus we can obtain $f_i(n)$ directly by adding $g_i(n)$ and $h_i(n)$ together. Finally, since the evaluation function f(n) defined before satisfies the minimax strategy, i.e.:

$$f(n) = \max_{1 \le i \le p} f_i(n)$$

our principle for expanding nodes in A^* search algorithm will obtain an optimal solution. Now we have shown how to solve our subproblem by using A^* search algorithm and minimax strategy.

V. Dynamic Routing Assignment Algorithm

After using workgraphs to represent tasks, we can use dynamic routing assignment algorithm to find a near optimal solution. In this section we will describe the details of our dynamic routing assignment algorithm in order to solve this routing assignment problem completely. This algorithm consists of the following two parts:

Part 1. If an AGV has just accomplished its assigned task and becomes free now, and if there are some other tasks ready to be processed, then we should assign this AGV to one of those tasks.

Part 2. After a task is chosen (so that the kind of this task is determined) for the AGV, we should determine a route for that AGV.

The two parts will be executed repeatedly till all the tasks are finished. Now we will describe how to proceed with each part to complete our dynamic planning and scheduling algorithm.

Suppose the number of every kind of tasks to be accomplished is initially N_1, ..., N_k as stated before. At some time later the number of every kind of tasks which have been accomplished becomes CN_1, ..., CN_k. Now if an AGV is freed and has not been assigned to any other task, then the problem is to choose one kind of uncompleted tasks for this AGV. To solve this problem, we first calculate the accomplishment ratio of all different kinds of tasks, that is to calculate all CN_i/N_i for all $1 \le i \le k$. then we choose one kind of tasks with the smallest accomplishment ratio for this AGV. If there are more than one kind of tasks which have the same ratio, then we can choose one of them randomly. Also note that if there are more than one AGV being freed at the same time, then we should use this heuristic rule repeatedly to assign tasks for those free AGV's.

After one kind of tasks for the free AGV has been chosen, the problem left is to choose a routing assignment for this AGV to accomplish the assigned task. In order to find a near optimal solution to accomplish all tasks, we must take all the uncompleted subtasks currently being taken care of by other AGV's into consideration. This means that if a busy AGV v_i is assigned a workgraph with vertices V_1, ..., V_{n_i}, and currently this AGV is performing the subtask in vertex V_{i_y} ($1 < i_y < m_i$), then we must consider all the uncompleted subtasks in vertices V_{i_y+1}, ..., V_{m_i} together in order to find a better routing assignment. So the predetermined route for AGV v_i to travel through vertices V_{i_y+1}, ..., V_{m_i} may be changed after a free AGV is added in to perform its newly assigned task. To be more specifically, suppose O^{i_y} is the predetermined routing assignment for AGV v_i to travel through vertices V_1, V_2, ..., V_{i_y}, and R^{i_y} is its corresponding travelling time history. Then the state description for AGV v_i is (i,R^{i_y},O^{i_y}) for now. Now, if there is an AGV, say, v_a, being freed now, after assigning this AGV to some kind of task, the state description of all AGV's become (a,ϕ,ϕ) plus (i,R^{i_y},O^{i_y}) for all busy AGV's v_i. And then we can use A^* search algorithm and minimax strategy described as before to find a near optimal solution. Also note that if there are more than one AGV being freed at the same time, say, v_a, v_b, ..., after assigning these AGV's to some kind of tasks, the current state description then becomes (a,ϕ,ϕ), (b,ϕ,ϕ), ..., plus (i,R^{i_y},O^{i_y}) for all busy AGV's v_i, so that the same algorithm can be used likewise. Because our approach repeatedly use A^* algorithm till all tasks to be finished, and the routing assignment for all AGV's change dynamically to meet the near optimality requirement, this is the reason why it constitutes a *dynamic routing assignment algorithm*. The more detailed, and complete steps of our method are described as follows.

Dynamic Routing Assignment Algorithm :

Step 1. At some time, for each free AGV, do the following steps:

Step 1.1. If there is no task to be assigned, then reset this AGV; otherwise goto Step 1.2.

Step 1.2. Compute the accomplishment ratio for all kinds of tasks uncompleted.

Step 1.3. Choose one kind of tasks with the smallest accom-

plishment ratio for this AGV to process. If there are more than one kind of tasks with the same smallest ratio, choose one of them randomly.

Step 2. If all tasks have been accomplished so that all free AGV's have no uncompleted task to be assigned, then exit this algorithm; otherwise goto Step 3.

Step 3. For every busy AGV v_i, do the following:

Step 3.1. Find the vertex V_{i_y} in its workgraph such that the AGV is currently doing a subtask contained in this vertex.

Step 3.2. Record the current travelling time history R^{i_y} and the current travelling workstation sequence O^{i_y} (up to the vertex V_{i_y}) for AGV v_i.

Step 3.3. Build the state description (i,R^{i_y},O^{i_y}) for the AGV v_i.

Step 4. Build the state description (a,ϕ,ϕ), (b,ϕ,ϕ), ..., for all free AGV's v_a, v_b, ..., which have been assigned some new tasks.

Step 5. Use the A^* algorithm as described previously to find a new routing assignments for all these AGV's.

Remark: If An AGV has just accomplished its assigned task and it is freed at a loading/unloading center, then we will use our dynamic routing assignment algorithm once to find new routes for all AGV's.

VII. Conclusion

In this paper, a new approach for solving the problem of planning and scheduling together was presented. This problem was first formulated as the determination of an optimal routing assignment for automated guided vehicles in an FMS. Later, we proposed a new approach that can solve this routing assignment problem dynamically. A computer simulation example is provided in [16] which shows a satisfactory result. In fact, the total computational time spent is also economical. The application of this method to an FMS in a real environment will be quite promising. Ongoing research will be on using more general representations for describing the class of decomposable tasks and better heuristics to improve the performance of the method.

References

[1] J. A. Simpson, R. J. Hocken, and J. S. Albus, "The automated manufacturing research facility of the National Bereau of Standards," *J. Manuf. Syst.*, vol. 1, pp. 17-32, 1982.

[2] R. Sui and C. K. Whitney, "Decision supporting requirements in flexible manufacturing," *J. Manuf. Syst.*, vol. 3, no. 1, pp. 61-69, 1984.

[3] P. E. Chen and J. Talavage, "Production decision support system for computerized manufacturing system," *J. Manuf. Syst.*, vol. 1, no. 2, pp. 157-168, 1982.

[4] C. K. Whitney, "Integration in FMS design and control," in *Proc. ASME Computer in Engineering Conf.*, pp. 355-360, 1986.

[5] R. E. Young, "Planning and control requirements for flexible manufacturing systems," in *Proc. ASME Computers in Engineering Conf.*, pp. 347-353, 1986.

[6] A. Ballakur and H. J. Steudel, "Integration of job shop control systems: A state-of-art review," *J. Manuf. Syst.*, vol. 3, no. 1, pp. 71-88, 1984.

[7] K. E. Stecke and T. L. Morin, "The optimality of balancing workload in certain types of flexible manufacturing system," *Europe. J. Oper. Res.*, vol. 20, pp. 66-72, 1985.

[8] Y. C. Ho and X. R. Cao, "Performance sensitivity to routing changes in queuing networks and flexible manufacturing systems using perturbation analysis," *IEEE J. Robotics Automation*, vol. RA-1, no. 4, Dec., 1985.

[9] O. Bermen and O. Maimon, "Cooperation among flexible manufacturing systems," *IEEE J. Robotics and Automation*, vol. RA-2, no. 2, pp. 24-30, Mar., 1986.

[10] B. R. Fox and K. G. Kempf, "Opportunistic scheduling for robotics assembly," in *Proc. IEEE Inter. Conf. on Robotics and Automation*, pp. 880-889, 1985.

[11] X. Xia and G. A. Bekey, "SROMA: An adaptive scheduler for robotic assembly system," in *Proc. IEEE Inter. Conf. on Robotics and Automation*, pp. 1282-1287, 1988.

[12] L. S. H. de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans," in *Proc. AAAI*, pp. 1113-1119, 1986.

[13] N. J. Nilsson, *Principle of Artificial Intelligence*, Palo Alto, CA: Tioga, 1980.

[14] C. C. Shen and W. H. Tasi, "A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion," *IEEE Trans. Comput.*, vol. C-34, pp. 197-203, Mar., 1985.

[15] C. L. Chen, C. S. George, and Clare D. Mcgillem, "Task assignment and load balancing of autonomous vehicles in a flexible manufacturing system," *IEEE J. Robotics and Automation* vol. RA-3, no. 6, Dec., 1987.

[16] P. S. Liu and L. C. Fu, "Planning and scheduling in a flexible manufacturing system using a dynamic routing assignment method for automated guided vehicles," Technical Report NTUCSIE 88-01, Department of Computer Science and Information Engineering, National Taiwan University, 1988.