Robotica (1991) volume 9, pp 393–408

Motion planning for multiple robots with multi-mode operations via disjunctive graphs*

Chi-Fang Lin† and Wen-Hsiang Tsai‡

(Received: November 12, 1990)

SUMMARY
A new approach to motion planning for multiple robots with multi-mode operations is proposed in this paper. Although sharing a common workspace, the robots are assumed to perform periodical tasks independently. The goal is to schedule the motion trajectories of the robots so as to avoid collisions among them. Rather than assigning the robots with different priorities and planning safe motion for only one robot at a time, as is done in most previous studies, an efficient method is developed that can simultaneously generate collision-free motions for the robots with or without priority assignment. Being regarded as a type of job-shop scheduling, the problem is reduced to that of finding a minimaximal path in a disjunctive graph and solved by an extension of the Balas algorithm. The superiority of this approach is demonstrated with various robot operation requirements, including "non-priority", "with-priority", and "multi-cycle" operation modes. Some techniques for speeding up the scheduling process are also presented. The planning results can be described by Gantt charts and executed by a simple "stop-and-go" control scheme. Simulation results on different robot operation modes are also presented to show the feasibility of the proposed approach.

KEYWORDS: Robot Operation; Motion Planning; Schedule Map; Disjunctive Graph.

1. INTRODUCTION

In a complicated automated industrial environment, there usually exists many kinds of moving robots programmed with specified functions, such as inspection, assembly, packing, etc. In such an environment, a planner to coordinate the activities of the robots to prevent collisions is desirable. For example, consider a flexible manufacturing factory equipped with numerous autonomous land vehicles (ALV's) which travel along prescribed routes to perform certain tasks repetitively. The fixed routes may be shared by multiple ALV's to utilize the workspace more efficiently. Collisions will be inevitable if their motions are not carefully scheduled in advance. As another example, cooperation of multiple robot manipulators usually is necessary in accomplishing a difficult task, such as assembling a complicated workpiece. Again, collisions can cause problems.

The above coordination problem is called the motion planning problem. In general, the motion planning problem is solved by first finding the safe path for each moving object among fixed obstacles (called the path planning problem), and next scheduling their motion along the paths for avoiding collisions among the objects (called the trajectory planning problem). The latter is the main topic studied in this paper. Proposed in this paper is a new approach to planning collision-free motion trajectories for multiple robots which work periodically and independently along fixed paths and share a common workspace.

Many previous approaches to motion planning concentrated on finding paths among fixed obstacles.[1–5] In general, they are not suitable for the problem mentioned above. In recent years, the problem of collision-free motion planning in a time-varying environment attracts more and more research interest. Some of the results are briefly reviewed below.

Lee and Lee[6] presented an approach to motion planning for two moving robots in a common workspace by the notions of collision map and time scheduling. Liu, et al.[7] proposed a method for planning collision-free coordinated motion of two mobile robots in the presence of obstacles. A two-level planner was developed and used to plan the motion in a Petri net created by the planner. Both approaches offer solutions to motion planning for two moving robots.

Some other studies proposed methods for planning a safe path for a single object among several moving obstacles. The methods can be modified to plan safe paths for a set of moving objects by first assigning different priorities to the objects and then planning the motion of the objects in a way of handling one object at a time in a priority order. Fujimura and Samet[8] solved the problem by including time as an additional dimension and treating the moving obstacles as stationary in an extended world model. A quadtree-type hierarchical structure is used to represent the model, and a collision-free path is planned without exceeding the prespecified range of velocity, acceleration, and centrifugal force. Kant and Zucker[9] decomposed the trajectory planning problem into two subproblems. The first is to plan a collision-free path for a moving object by ignoring the other moving obstacles, and the second is to plan a velocity schedule along that path while avoiding

* This work was supported partially by National Science Council, Republic of China under grant NSC79-0404-E009-17.
† Institute of Computer Science and Information Engineering.
‡ Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050 (Republic of China).

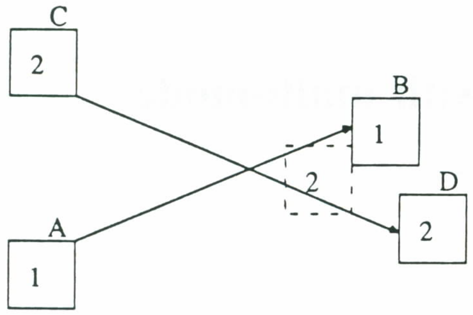


Fig. 1. A motion planning problem of two moving objects.

collisions with the obstacles. Erdmann and Lozano–Pérez[10] presented another approach to motion planning for multiple moving objects. A configuration space is created each time when a certain moving object changes its velocity, and the configuration space-time of the object is represented as a series of two-dimensional configuration space slices at different time instances. After defining a visibility graph for the configuration space-time, a collision-free path is searched from the initial slice to the final one using graph search algorithms.

The above priority-based planning methods, though efficient, do not consider all the moving objects simultaneously when planning the trajectories. A planner implemented by these methods might design a trajectory for an object with higher priority and fail to find a solution for another object with lower priority. A simple example of such motion planning for two square-shaped objects is shown in Figure 1, in which object 1 with higher priority is scheduled first and reaches the goal position *B* before object 2 is scheduled, which needs to reach the position *D*. In this case, no solution can be found for object 2 because object 2 always hits object 1 before reaching *D*. The C-space approach mentioned previously can be extended conceptually to solve this problem by planning the motion in a high dimensional configuration space,[10] e.g. by planning the motion of n objects each with k degrees of freedom in an $n \times k$ dimensional space. However, the complexity of the extended method is greatly increased if a large number of moving objects need be handled.

A new approach to planning safe motion for multiple robots with or without priority assignment is proposed in this paper. The motion is planned in such a way that all the moving robots are considered and scheduled simultaneously. The concept of schedule map introduced in ref. 11 is used, and the planning problem is reduced to that of finding a minimaximal path in a disjunctive graph by the assistance of the maps. The Balas algorithm[12] accompanied by a process of validity checking is employed to plan collision-free motion schedules for the robots with various operation modes. Some techniques for speeding up the process of planning are also presented. The scheduling results can be represented as Gantt charts, and it is easy to execute by a controller by simply issuing a sequence of "STOP" and "GO" control signals to the robots.

The major contributions of this study include the following:

(i) The problem of motion planning for multiple robots, which is believed hard to solve, is reduced to a path-finding problem in a disjunctive graph which can then be solved by traditional approaches like the Balas algorithm.

(ii) The techniques for implementing a motion planner are developed. The planner can be used to schedule the motion of multiple robots under the requirements of various robot operation modes.

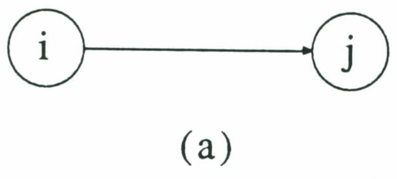
(iii) The complexity of the proposed methods is not greatly increased when a large number of robots are scheduled; the scheduling results can be obtained in a reasonable amount of time and are easy to execute.

The paper is organized as follows. Section 2 includes an overview of the basic concepts of the proposed approach. In Section 3, the Balas algorithm is briefly reviewed first, followed by the presentation of the proposed planning method. Collision-free motion planning for various robot operation modes are also discussed. In Section 4, some methods for speeding up the planning process are presented. Simulation results are given in Section 5. Discussions and conclusions are presented finally in Section 6.

2. BASIC IDEA OF MULTI-ROBOT MOTION PLANNING VIA DISJUNCTIVE GRAPHS

The objective of this study is to develop techniques for coordinating the motion of multiple robots before they are actually put into operations so that collisions among the robots can be avoided. The robots are assumed to work periodically and independently and share a common workspace. The working path of each robot is programmed independently in a teaching stage to complete a certain task and is not changed thereafter. Overlapping of robot paths is possible because the workspace is shared by multiple robots, so collisions may occur if the robot operations are not carefully scheduled. Thus it is desired to avoid collisions among the robots by planning their motion in advance. This can be regarded as a problem similar to job-shop scheduling in which space is the shared resource.

The job-shop scheduling problem[12–17] is a problem of finding an optimal processing sequence for a set of jobs on a set of machines such that the completion time for all the jobs is minimized under the following constraints: (1) each job is processed on a set of machines in a prespecified order; (2) no more than one job is allowed to be performed on a single machine at any time; and (3) once an operation pertaining to a certain job is executed on a particular machine, it may not be interrupted by another operation until it is completed on that machine, i.e., a nonpreemptive constraint is enforced. It has been shown that the job-shop scheduling problem is NP-complete just like the traveling salesman problem.[18,19] So, a good approximate solution instead of the best one usually is recommended in most studies on job-shop scheduling. Many previous researchers formulated this type of problem as one of finding a



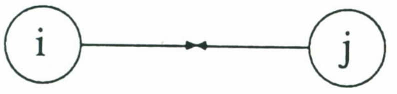


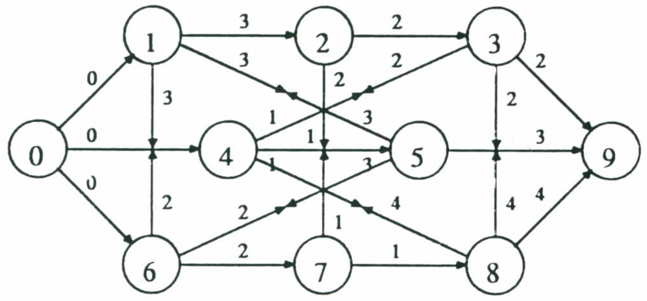
Fig. 2. Graphic representations of arcs. (a) A conjunctive arc. (b) A pair of disjunctive arcs.

minimaximal path in a disjunctive graph.[12–15] A brief description of this graph-theoretic formulation and a simple illustrative example are given below.

Let $\{i \mid i=1, 2, \ldots, J\}$ be a set of jobs in which job i and n_i operations indexed from $(\sum_{l=1}^{i-1} n_l)+1$ to $\sum_{l=1}^{i} n_l$ (specifically, job 1 has operations indexed from 1 to n_1, job 2 has operations indexed from n_1+1 to n_1+n_2, and so on), and $\{k \mid k=1, 2, \ldots, M\}$ be a set of machines. Also let $N=\sum_{i=1}^{J} n_i$ and $n=N+1$. A disjunctive graph is a directed graph defined as $G=(\mathscr{V}; \mathscr{C}, \mathscr{D})$, where (1) $\mathscr{V}$ is a set of nodes including N nodes corresponding to all the N operations pertaining to the jobs as well as two additional dummy nodes, node 0 and node n, indicating the source and the sink of the graph, respectively; (2) $\mathscr{C}$ is a set of conjunctive arcs with each element (i, j) representing the operations pertaining to an identical job in the prespecified order; $\mathscr{C}$ also includes, for each job k, two additional arcs $(0, i_k)$ and (j_k, n) where i_k and j_k are the first and the last operations of job k, respectively; (3) $\mathscr{D}$ is a set of disjunctive arc pairs with each pairs of arcs (i, j) and (j, i) representing the operations pertaining to different jobs but performed on a single machine. For any arc pair (i, j) and (j, i) in $\mathscr{D}$, (j, i) is termed as the complement of (i, j). Associated with each arc (i, j) in $\mathscr{C} \cup \mathscr{D}$ is the processing time $d(i)$ of operation i. The time $d(0)$ associated with each arc $(0, i)$ is defined to be zero. A conjunctive arc in $\mathscr{C}$ is depicted as an arc in a single direction, and a pair of disjunctive arcs in $\mathscr{D}$ as two arcs in contact with opposite directions as shown in Figure 2. A simple example of a (3×3) job-shop problem and its disjunctive graph representation are shown in Figure 3.

Job (i)	1	1	1	2	2	3	3	3
Operation (j)	1	2	3	4	5	6	7	8
Machine (k)	1	2	3	3	1	1	2	3
Duration (d(j))	3	2	2	1	3	2	1	4

(a)



(b)

Fig. 3. A graphic representation of a job-shop problem. (a) A (3×3) job-shop problem. (b) The corresponding disjunctive graph.

A feasible schedule for a given job-shop problem, which is represented by a disjunctive graph G defined above, can be obtained by selecting one arc from each pair of arcs in D such that the resulting graph does not contain any cycle (i.e. the graph is circuit-free). Let $\mathscr{F}=\{F_1, F_2, \ldots, F_w\}$ be the set of feasible schedules for G and $\mathscr{G}=\{G_1, G_2, \ldots, G_w\}$ be the set of circuit-free graphs corresponding to $\mathscr{F}$. A path from node 0 to node i in a graph is defined as a sequence of arcs, and the length of a path is defined as the sum of the processing times associated with the arcs in the path. The length of the longest path from node 0 to node i, which is denoted as C_i, in a circuit-free graph G_h in $\mathscr{G}$ is computed as follows:

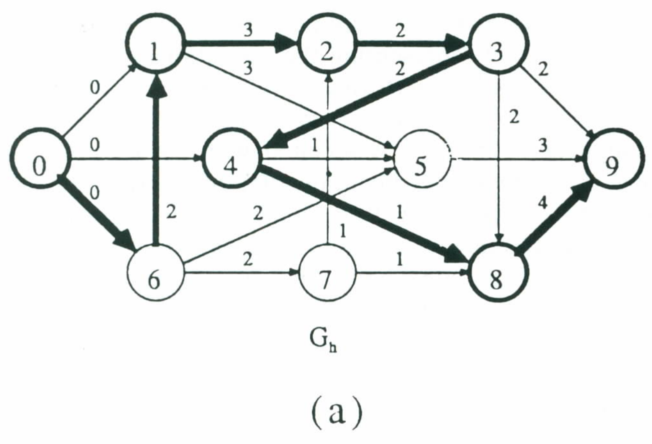
$$C_i = \max_{\forall j,(j,i) \text{ is an arc in } G_h} (C_j + d(j)), \qquad C_0 = 0 \tag{1}$$

where $d(j)$ represents the processing time of j. The longest path from source to sink in G_h, which may not be unique, is called a critical path in G_h. The completion time of a feasible schedule F_h can be seen as the length of a critical path in G_h (i.e. C_{n_h} in equation (1) if n_h is the sink of G_h), and is called the cost of the schedule and the corresponding graph henceforth. Also, we define the optimum schedule to be the schedule with the minimum cost. As an example, a feasible schedule for the problem depicted in Figure 3 is shown in Figure 4. The critical path is shown by the bold lines, and the cost is 14. A critical path in G_k is called a minimaximal path (a path with the maximal length from source to sink and the minimum cost) in G if the corresponding feasible schedule F_k is the optimum schedule with the minimum cost

$$C_{n_k} = \min_{\forall G_h \in \mathscr{G}} C_{n_h} \tag{2}$$

where n_i represents the sink of graph G_i. Thus, it is easy to see that the problem of job-shop scheduling is equivalent to that of finding a minimaximal path in a disjunctive graph. This concept is employed in this study to solve the problem of motion planning. The principle is described briefly below; the details are described in the next section.

Assume that r robots $(r>2)$ indexed from R_1 to R_r are considered in the motion planning procedure. The first step is to construct a schedule map $MAP_{p,q}$ for each pair of robots (R_p, R_q) using the method presented in ref. 11



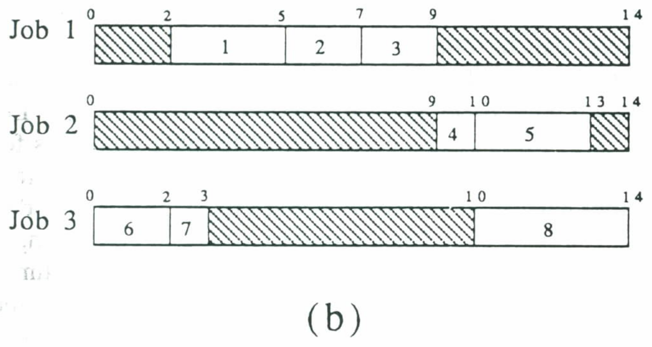
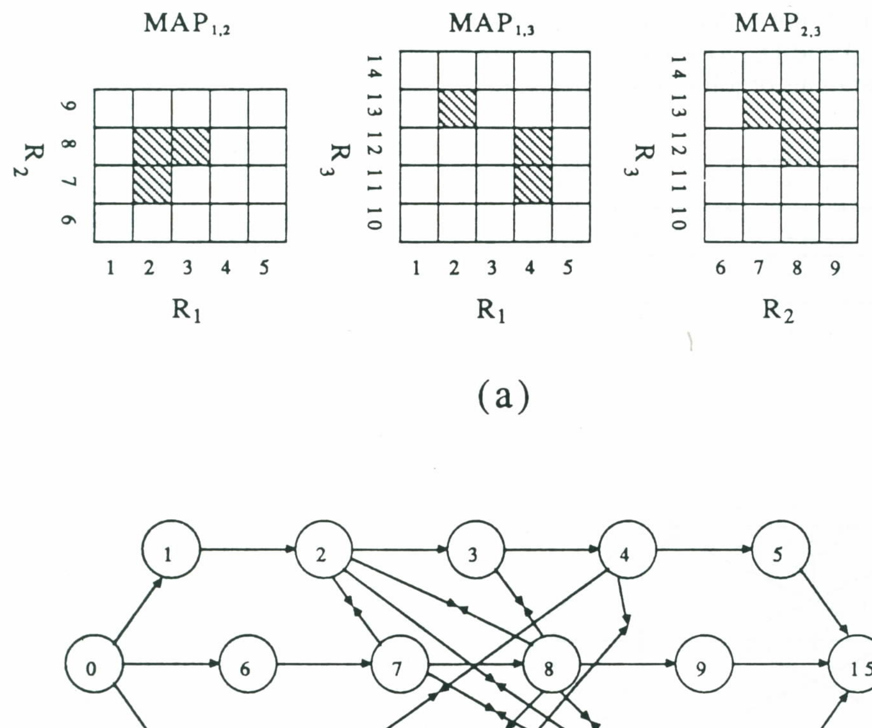


Fig. 4. A feasible schedule for the problem shown in Figure 3. (a) A circuit-free graph which represents a feasible schedule. (b) The feasible schedule represented by a Gantt chart.

where $1 \leq p < q \leq r$. The schedule map is a two-dimensional figure with the horizontal axis representing the execution time of robot R_p to complete a periodical task and the vertical axis representing that of the other robot R_q to complete another periodical task, both of which can be obtained in the teaching stage. The axes are further decomposed into two series of small time intervals, each with a constant time value h (i.e. the length of each axis is assumed to be composed of a multiple of h). Accordingly, the map looks like a chessboard with the resolution determined by the numbers of the time intervals specified for the two robots. The task performed on each robot conceptually can be divided into a series of pseudo subtasks corresponding to the time intervals on the axis. Each subtask requires an identical amount of processing time. The completion of the task is achieved by accomplishing the corresponding sequence of pseudo subtasks. See Figure 5(a) for an example. Some squares in the maps, which are shown shaded in Figure 5(a), indicate that collisions will occur if both robots perform the pseudo subtasks of the corresponding time intervals simultaneously. They are termed as collision squares in the sequel. Two additional processes, namely, the trajectory modeling process to model the trajectory information of a moving robot and the collision detection process to detect possible collisions between two moving robots using the modeling results, are required to determine the



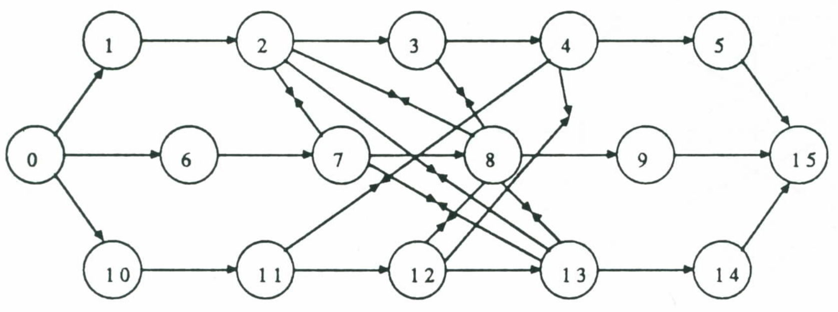


Fig. 5. A graphic representation of the multi-robot scheduling problem. (a) Schedule maps for the three robots R_1, R_2, and R_3. (b) The corresponding disjunctive graph.

collision squares in the map. The details can be found in ref. 11.

By regarding each robot task as a job and the sequence of the constituting pseudo subtasks of the task as the operations pertaining to the job, the problem of multi-robot motion planning can be formulated as a job-shop scheduling problem, which can then be represented by a disjunctive graph. Each subtask is represented by a graph node, and each collision square is represented as a pair of disjunctive arcs that join the two nodes (subtasks) corresponding to the two time intervals spanning that square. The graphic representation of the example shown in Figure 5(a) is depicted in Figure 5(b).

3. MOTION PLANNING FOR MULTIPLE ROBOTS WITH MULTI-MODES

The appeal of the disjunctive graph representation scheme for the multi-robot motion planning problem is that we can plan collision-free motion schedules for multiple robots by finding a minimaximal path in the graph under certain conditions. Collision-free motion planning for various robot operation modes can also be achieved. The details are discussed in this section.

3.1 Proposed method for planning collision-free motion schedules

The enumeration algorithm presented by Balas[12] for finding a minimaximal path in a disjunctive graph is briefly reviewed first. The Balas algorithm solves the problem by generating a sequence of circuit-free graphs and maintaining a search tree. Each graph G_s in the sequence is obtained from some previously generated graph G_p by complementing one of the disjunctive arcs (i, j) (i.e. by replacing arc (i, j) with arc (j, i)) which are on a critical path in G_p. Whenever a new graph G_s is generated with G_p as the preceding graph, a new node G_s and a new arc (G_p, G_s) indicating that G_s is a descendant of G_p and G_p is an ancestor of G_s are added to the search tree. The complemented arc (i.e. arc (j, i)) in the newly created graph is then fixed and called a fixed arc[12] in the sense that it cannot be complemented any more in this new graph, and in any of the descendants of the graph in the search tree. The next possible candidates to be complemented are those disjunctive arcs which are not fixed yet and are on a critical path in the current graph. By using the minimum cost among all the graphs generated so far as the current upper bound and the cost of the current partial graph containing only the fixed arcs and the conjunctive arcs as the lower bound, a branch-and-bound technique is employed to reduce the number of nodes generated in the tree. If the lower bound computed in G_s is greater than or equal to the current upper bound, then graph G_s with all its potential descendants in the search tree is abandoned, and the search is backtracked to the graph from which G_s was generated (because generating the tree further cannot bring any improvement on the upper bound). When backtracking from G_s to G_p, arc (i, j) in G_p is also fixed if G_s was previously generated from G_p by complementing arc (i, j).

The Balas algorithm starts with any graph that contains a feasible solution corresponding to a feasible schedule as the root node in the search tree, and gradually improves the feasible solution by the above procedure. The algorithm terminates when backtracking is necessary for the root node. The optimal sequence of operations is then given by graph G^* which contains the minimum cost among all the graphs in the search tree, and any critical path in G^* is called the minimaximal path in G. A possible root node of the search tree is the graph containing only the conjunctive arcs and the normal disjunctive arcs (arc (i, j) in $\mathscr{D}$ is called normal if $i < j$). A simple example is illustrated in Figure 6 with critical paths shown as bold lines. Other related but different approaches can be found in refs. 13, 14, 17.

The Balas approach is not directly applicable to motion planning of multiple robots. Actually, the optimal schedule obtained by the Balas approach may be an unsafe motion schedule for multiple robots. Before the reason is given, we first introduce the notion of schedule paths on schedule maps and next define valid and invalid schedule paths.

As mentioned previously, the work path of each robot for a particular task is taught in the teaching stage and not allowed to change thereafter. However, the trajectory information of each robot path can be modified by inserting waiting times into the original time schedule to prevent possible collisions. The following two assumptions are made in this study for easier implementation of a motion planner:

(1) the time instances at which waiting times can be inserted are the ends of the time intervals on the axes of the collision maps; insertions in between are not allowed (i.e. the subtasks are performed in a nonpreemptive manner); and

(2) the length of each inserted waiting time duration is restricted to a multiple of h, which is a time interval to process a subtask.

Based on these assumptions, a schedule path $L_{p,q}$ on map $MAP_{p,q}$ can be represented by any line as follows:

(1) non-decreasingly coming from the lower left corner of the map to the upper right corner; and

(2) being composed of three basic types of line segments with horizontal (processing the corresponding subtask on the horizontal axis only), vertical (processing the corresponding subtask on the vertical axis only), and 45° (simultaneously processing the corresponding subtasks on both axes) directions.

A simple example is shown in Figure 7(a).

The method of constructing $L_{p,q}$ on $MAP_{p,q}$ based on the precedence relations described in a circuit-free graph G_s is presented as follows. The path is constructed from the lower left corner of $MAP_{p,q}$ to the upper right corner by piecewisely adding a series of basic line segments. Assume that square (i, j) is in consideration where i is on the horizontal axis of the map and j is on the vertical axis, and the corresponding subtasks are performed on R_p and R_q, respectively. The method includes the following three steps:

1. if i is a predecessor of j (i.e. if there exists a path

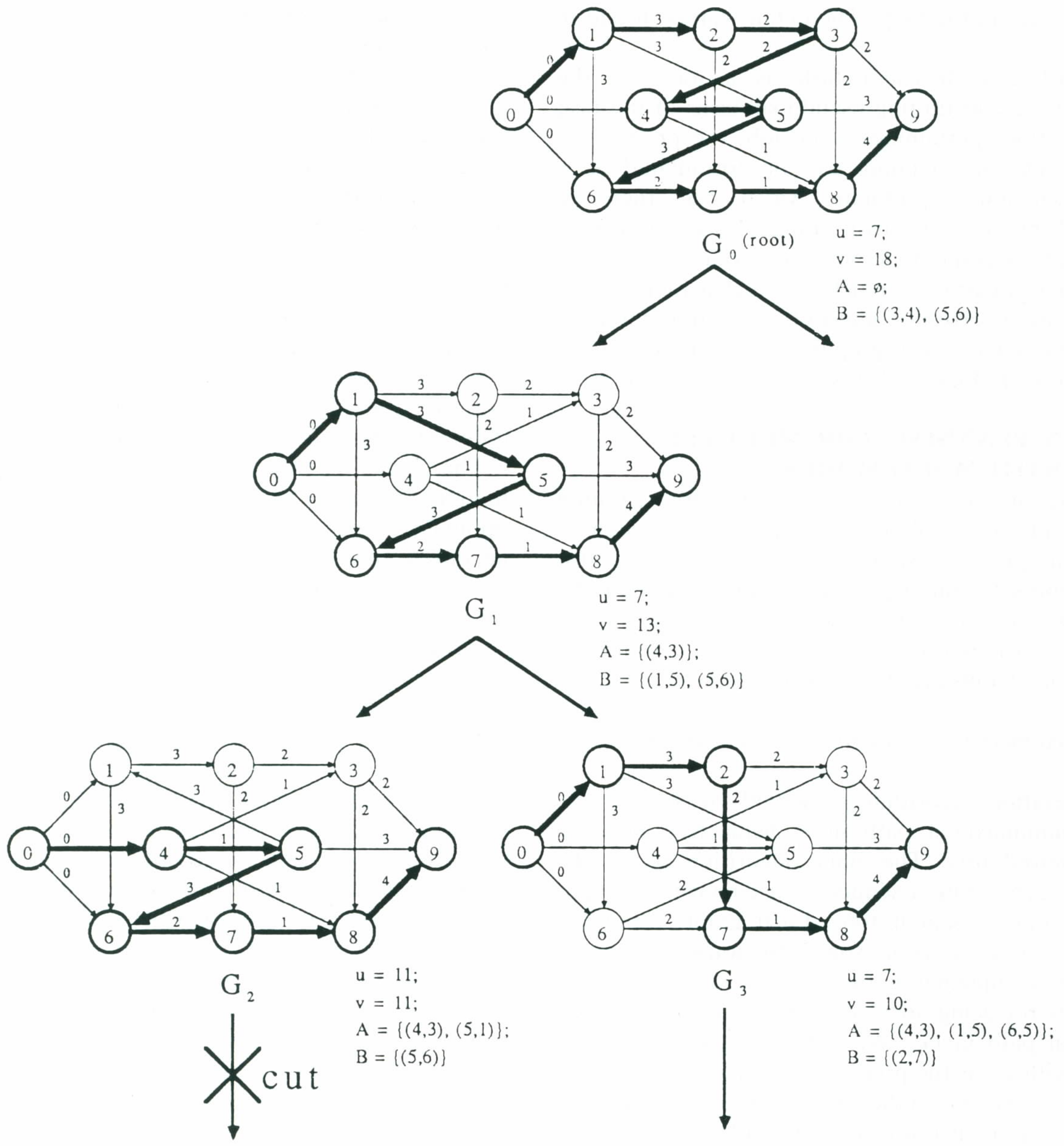


Fig. 6. A simple example to illustrate part of the search tree generated by the Balas algorithm (*u* is a lower bound; *v* is an upper bound; *A* is a set of the fixed arcs; and *B* is a set of candidates to be complemented).

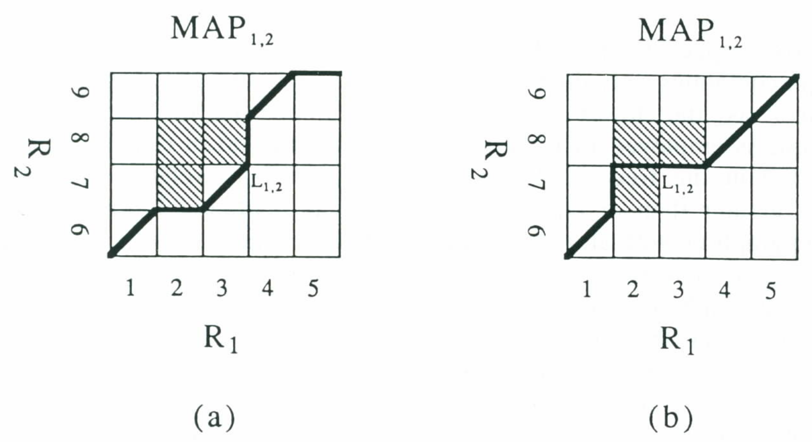


Fig. 7. A valid and invalid schedule paths planned on a schedule map. (a) A valid schedule path. (b) An invalid schedule path.

from node i to node j), add a horizontal line segment along the bottom edge of square (i, j) to $L_{p,q}$ and continue the process with square $(i+1, j)$;
2. if j is a predecessor of i, add a vertical line segment along the left edge of square (i, j) to $L_{p,q}$ and continue the process with square $(i, j+1)$; and
3. if i and j are not predecessors of each other, add a 45° line segment diagonally over square (i, j) to $L_{p,q}$ and continue the process with square $(i+1, j+1)$.

Note that the condition that both i and j are predecessors of each other is impossible because G_s is circuit-free in nature. On the other hand, each set of adjacent collision squares are grouped together in advance to form a collision region. Any schedule path that enters the interior of a collision region is not allowed, and it is termed as an invalid schedule path. This is called the interference constraint in this study. Although the graphs generated by Balas are guaranteed circuit-free and provide feasible schedules for a given job-shop problem as mentioned in Section 2, their corresponding schedule paths may possibly violate the interference constraint, resulting in unsafe motion schedules. To avoid this problem, we treat each collision region as a polygon and check the interference constraint for each newly added line segment when constructing the schedule path. Relevant methods can be found in Boyse.[20] See Figure 7(b) for an example.

Finally, the overall motion planning procedure for multiple robots is stated as follows. The Balas algorithm is employed and a sequence of circuit-free graphs is generated. A search tree is also maintained. Whenever a new graph is created at any stage of the Balas algorithm, the related schedule paths are constructed first using the precedence relations described in the new graph as well as the method presented above (including validity checking). If none of the schedule paths is checked to be invalid, the lower bound of the graph is then computed and tested against the current upper bound for the tree pruning purpose; otherwise, none of the bounds is computed and compared. The search tree is updated and the next graph is generated. The process is continued until the termination of the Balas algorithm. Finally, the operation sequence given in G^* is taken to be the optimum collision-free motion schedule for the robots.

3.2 Planning for different robot operation modes

The following three robot operation modes are studied in this section: (1) the "non-priority" operation mode; (2) the "with-priority" operation mode; and (3) the "multi-cycle" operation mode. Before presenting the methods, various variables are introduced first.

For any nodes i and k in a certain graph G_s created from the Balas algorithm, we define k to be the direct predecessor of i if (k, i) is an arc in G_s. We also define the critical direct predecessor of i to be the direct predecessor which is on the longest path from node 0 to node i. Some notations used in the following discussions are listed below:

$\mathscr{P}_i$:	the set of critical direct predecessors of i;
$\mathscr{R}_i$:	the set of robots on which the subtasks corresponding to nodes in $\mathscr{P}_i$ are performed;
$r(i)$:	the robot on which the subtask corresponding to i is performed
$s(i)$:	the start time of i;
$e(i)$:	the end time of i;
$d(i)$:	the processing time of i; and
$t(i)$:	the tardiness of i.

The tardiness $t(i)$ is defined to be the time interval between the end time of k which is the direct predecessor of i in the prespecified order (i.e. $r(i)$ and $r(k)$ specify the same robot) and the start time of i, i.e.

$$t(i) = s(i) - e(k). \tag{3}$$

Set $\mathscr{P}_i$ for each node i can be obtained by collecting those nodes which satisfy the constraint of the right hand side of equation (1) in computing the length of the longest path from node 0 to node i. The methods for planning collision-free motion schedules for robots under the requirements of these operation modes are presented below.

In the "non-priority" operation mode (mode 1), none of the robots is given the privilege to move in a higher priority than the other robots. The planning problem can be solved by using the method presented in the last section. After the termination of the Balas algorithm, an optimal planning result described in G^* is obtained. The stop-and-go control scheme (i.e. a sequence of "STOP" and "GO" control signals for each robot) can be easily obtained by finding $s(i)$ and $e(i)$ for each node i in G^*. Assume that node j is in $\mathscr{P}_i$. Then the values $s(i)$ and $e(i)$ are determined as follows:

$$s(i) = e(j) \tag{4}$$

and

$$e(i) = s(i) + d(i), \tag{5}$$

with the initial conditions being $s(0) = 0$ and $e(0) = 0$. The way of choosing an initial graph as the root node of the search tree will be discussed in the next section.

The second case is the "with-priority" operation mode (mode 2) where different priorities are given to the robots. The planner implemented under this operation mode must issue a "STOP" command to hold a robot with a lower priority to prevent any conflict with the robot with a higher priority (i.e. the latter is given the privilege to move first). The planning method of this mode is similar to that of mode 1, except that an additional priority constraint for each graph in the search tree is enforced to check if the requirement of the "with-priority" is satisfied. The method is stated below.

If the tardiness of a certain node i, i.e. $t(i)$, is found to be greater than 0, then idleness of robot $r(i)$ will occur because the start time of i is not identical to the end time of k which is the direct predecessor of i in the prespecified order. In this condition, $r(i)$ will be held to wait for the completion of each node in $\mathscr{P}_i$ before it can start to process i. Moreover, according to the requirement of the priority constraint, idleness of $r(i)$ is said to be invalid if none of the robots in $\mathscr{R}_i$ owns a priority higher than or identical to that of $r(i)$.

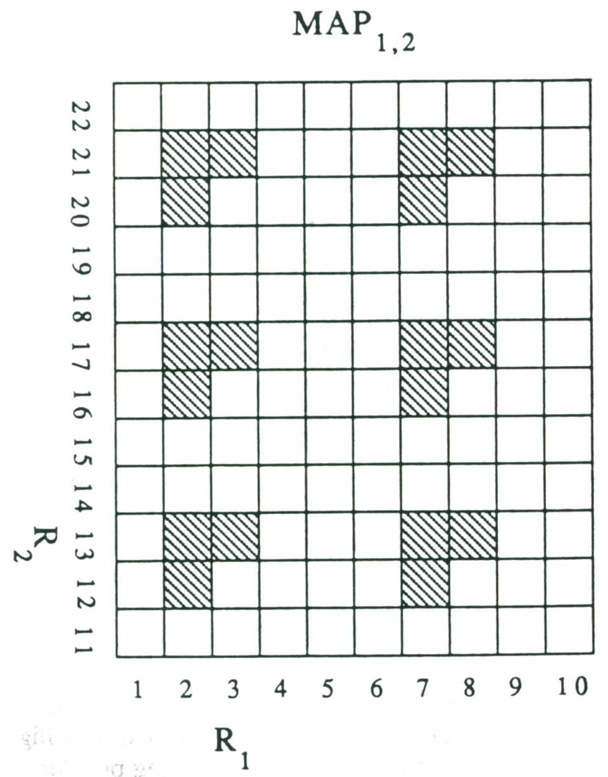


Fig. 8. An enlarged schedule map.

Therefore, by finding $\mathcal{P}_i$ and the corresponding $\mathcal{R}_i$ and checking $t(i)$ for each node i in a certain graph, the priority constraint can be verified for this graph. Only graphs satisfying the interference constraint as well as the priority constraint are useful in our planning procedure for this mode.

The last case is the "multi-cycle" operation mode (mode 3). The planning results of mode 1 and mode 2 discussed above are all based on the assumption that only "one-cycle" operation is considered, which means that if a robot has finished its own task, it must wait until the other robots also complete their tasks before the next cycle can be started. This restriction can be removed by duplicating the original maps of the robots along the horizontal and vertical directions. For example, an enlarged schedule map $MAP_{1,2}$ for a "2-cycle" operation of R_1 and a "3-cycle" operation of R_2 are depicted in Figure 8. The methods mentioned above can then be applied directly to such enlarged maps to get "multi-cycle" planning results.

4. IMPROVING PROPOSED PLANNING METHOD

Two important issues in the proposed approach that are related to both of the optimal solution and the searching time needed for obtaining the solution are discussed in this section. The first issue is how to select a graph as the root node of the search tree, and the second issue is how to reduce the number of disjunctive arcs in the graph.

4.1 Selection of initial graphs

In defining the initial graph $G = (\mathcal{V}; \mathcal{C}, \mathcal{D})$, some heuristic rules are found useful in selecting the directions of the arcs in $\mathcal{D}$. The rules can be used to make decisions according to the priority sequence and task completion time. For each pair of the disjunctive arcs, the rules are as follows.

(i) In mode 1, we choose the arc that points from a "longer-completion-time" node (i.e. a node pertaining to a robot which needs more time to complete a prescribed task) to a "shorter-completion-time" node; if a tie exists, we choose the normal one.

(ii) In mode 2, we choose the arc that points from a "higher-priority" node (i.e. a node pertaining to a robot with a higher priority) to a "lower-priority" node; if both nodes belong to the robots with an identical priority, then choose the arc according to Rule 1.

The root graph generated by the above rules has the following properties: (1) it is obviously circuit-free, and its corresponding schedule paths satisfy the interference constraint because the disjunctive arcs are chosen in a consistent manner; and (2) it satisfies the priority constraint in operation mode 2 because a high priority robot is always given the privilege to work first when conflicting with a low priority robot. So, the selected graph contains a rough but safe motion schedule for the robots. On the contrary, if the disjunctive arcs are selected randomly, possibly violating the rules in 1 and 2, the graph may contain an unsafe and useless schedule. Computation time will so be wasted or a safe but more costly schedule will result. For example, if the reverse arc of Rule 1 is selected, although the robot performing the short task finishes its own work first, it still needs to wait for the completion of the tasks performed on the other robots before it can start to execute the next cycle. So, the cost is increased because a robot performing the long task must be stopped when conflicting with the robot performing the short task.

4.2 Reduction of arcs in disjunctive graphs

The second issue related to the efficiency of the proposed approach is how to reduce the number of disjunctive arcs involved in $\mathcal{D}$. A graph with a large set $\mathcal{D}$ usually needs more computation time to find the optimal solution than a graph with a small set $\mathcal{D}$. As mentioned in Section 2, the size of $\mathcal{D}$ is determined by the number of collision squares found in schedule maps. However, some squares in the maps, as found in this study, are redundant for planning safe motion schedules for multiple robots so that the corresponding disjunctive arcs can be deleted from $\mathcal{D}$ without changing the final planning result. To find the redundant squares, the following method is proposed.

First, a rectangle in a map is designed to fully or partially cover a collision region by specifying a pair of collision squares A and B in the region, which are in the upper left and lower right positions of the rectangle. The rectangle can be a horizontal or vertical one-square thick bar as shown in Figure 9. Next, it can be shown that all collision squares grouped in the region and covered by the rectangle except A and B are actually redundant.

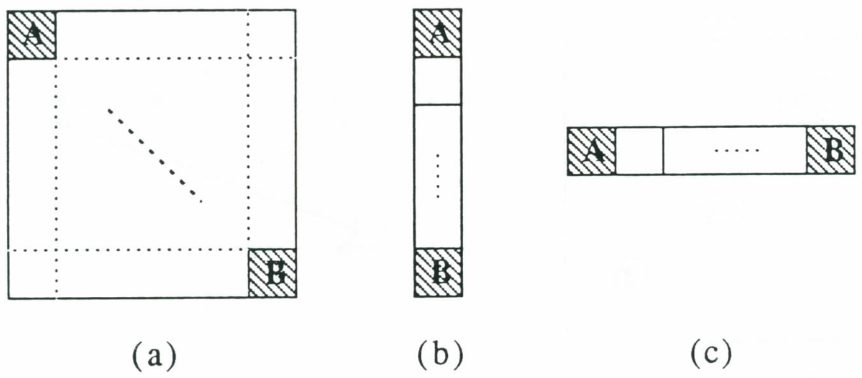


Fig. 9. Rectangles designed for detecting redundant arcs. (a) A rectangle specified by two corner squares A and B. (b) A vertical one-square thick rectangle. (c) A horizontal one-square thick rectangle.

Before the details are given, a fact stated below is proved first.

Assume that square (i, j) in a certain map is a collision square, and subtask i precedes subtask j in order. It can be proved that the schedule path in the map cannot penetrate into the shaded rectangular area (a, b, c, d) as shown in Figure 10(a). The proof is given as follows. Because i precedes j in order, all the predecessors of i must also precede j for otherwise a circuit is trivially formed. Therefore, whenever a certain square on the bottom side of the area, namely, square (i', j), is considered in constructing the schedule path with i' being the predecessor of i, a horizontal line segment along the bottom edge of square (i', j) is added according to the rules presented in Section 3 (also see Figure 10(a)). Moreover, because the schedule path is a non-decreasing curve connecting the lower left corner of the map to the upper right corner as defined in Section 3, it cannot "turn back" and penetrate into the right side of the area. Thus, it is concluded that the schedule path will not enter the interior of the rectangular area under this assumption. The case with the assumption that subtask j precedes subtask i in order is also depicted in Figure 10(b). The proof for showing redundant collision squares is now given below.

Assume that the two corner squares A and B are (i', j''), and (i'', j'), respectively. For any square (i, j) in the rectangle specified by A and B as shown in Figure 11(a), if (i, j) is a collision square, only one of the corresponding pair of disjunctive arcs (i, j) and (j, i) is selected to form a circuit-free graph as stated in Section 2. Assume that arc (i, j) instead of (j, i) is selected (this means that subtask i precedes subtask j in order). Then, based on this assumption and the fact proved above, we claim that arcs (i', j'') and (i'', j') must also be selected for the other two collision squares to obtain a valid schedule path. Part of the corresponding graph is shown in Figure 11(b), and the reason is stated as follows:

1. if arc (j'', i') instead of (i', j'') is selected (i.e. j'' precedes i' in order), then none of the schedule path can be obtained without entering the two overlapping shaded areas as shown in Figure 12(a), and the resulting graph is no more circuit-free in this condition because a circuit containing nodes i, j, j'', and i' is formed; and
2. if arc (j', i'') instead of (i'', j') is selected (i.e. j' precedes i'' in order), it can be seen from Figure 12(b) that the schedule path not entering the two separated shaded areas will enter the interior of the collision region formed by the three collision squares and is thus invalid.

Accordingly, the precedence relations described in the

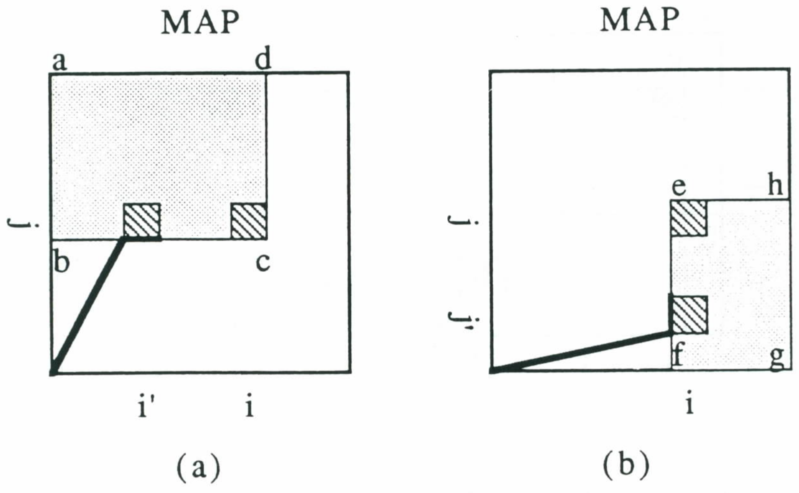


Fig. 10. Two shaded rectangular areas through which the schedule path cannot pass. (a) Area (a, b, c, d) corresponding to arc (i, j). (b) Area (e, f, g, h) corresponding to arc (j, i).

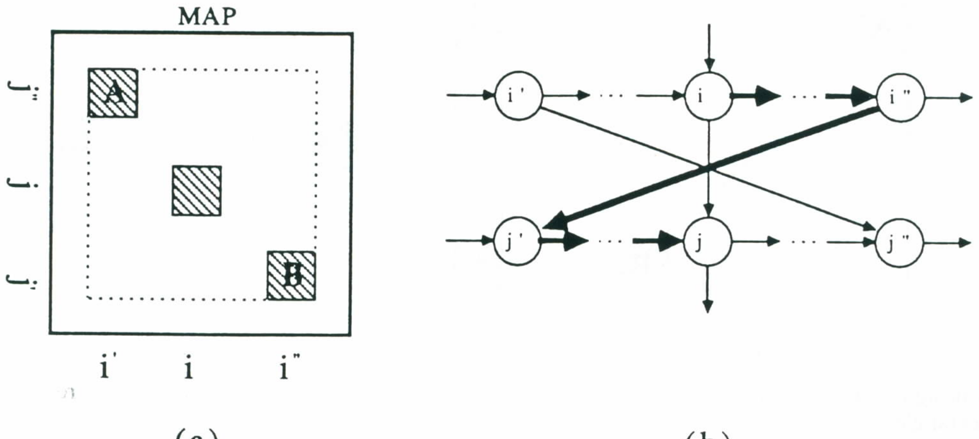


Fig. 11. Square (i, j) is a collision square and can be shown as redundant. (a) The rectangle containing square (i, j) and specified by A and B. (b) The partial graph corresponding to the three collision squares.

graph of Figure 11(b) are necessary for obtaining a valid schedule path. From this picture, it is noted that arc (i, j) is not an arc on any critical path in the graph because any path containing (i, j) is always shorter than the path containing the bold arcs. Therefore, the cost of the graph, which is just the total processing times associated with the arcs in a critical path, is not changed if (i, j) is deleted. Similarly, we can prove that arc (j, i) is redundant under the assumption that (j, i) instead of (i, j) is selected (this means that subtask j precedes subtask i in order). The details are omitted, and this completes the proof. The other cases of the horizontal and vertical rectangular bars can also be proved similarly.

According to the foregoing discussion, the method finally can be described as follows: select pairwisely collision squares A and B in a collision region and in the upper left and lower right positions of a rectangle of any size, and mark as redundant those collision squares except A and B. The remaining non-redundant squares are called guiding squares in this study.

It should be noted that although the redundant arcs can be deleted from $\mathscr{D}$ without changing the planning result as discussed above, the corresponding redundant collision squares are still necessary for checking whether the constructed schedule paths violating the interference constraint or not and should not be removed from the maps. The effectiveness of the improved planning method will be illustrated by several simulation experiments given in the next section.

5. SIMULATION RESULTS

Three robots (R_1, R_2, and R_3) are used and scheduled in our simulation experiments. The operation times for R_1, R_2, and R_3 to complete prescribed periodical tasks are assumed to be $10h$, $14h$, and $17h$, respectively, where h is a constant value. In operation mode 2, the priority sequence in ascending order for the three robots is (R_2, R_3, R_1). A "2-cycle" operation for R_1 and "1-cycle" operation for both R_2 and R_3 are planned in operation mode 3. The collision maps $MAP_{1,2}$, $MAP_{1,3}$, and $MAP_{2,3}$ are manually constructed for demonstration as

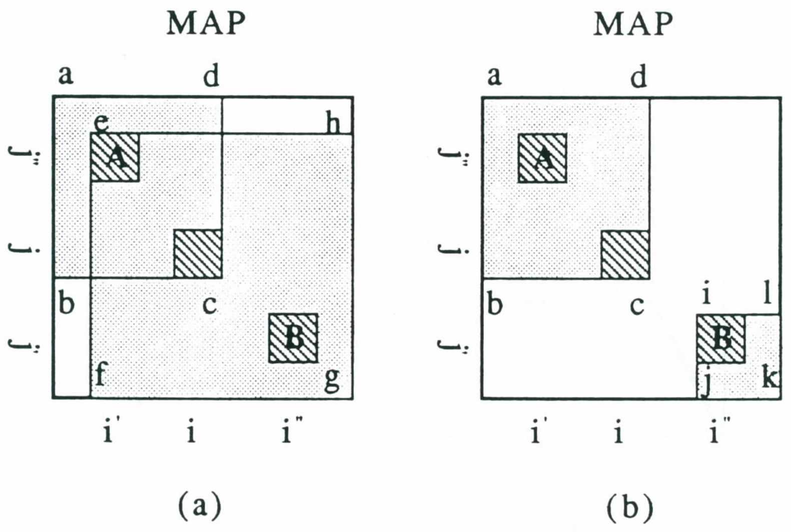


Fig. 12. The schedule path cannot enter the interior of the shaded areas. (a) Two over-lapping rectangular areas (a, b, c, d) and (e, f, g, h) corresponding to arcs (i, j) and (j'', i'), respectively. (b) Two separated rectangular areas (a, b, c, d) and (i, j, k, l) corresponding to arcs (i, j) and (j', i''), respectively.

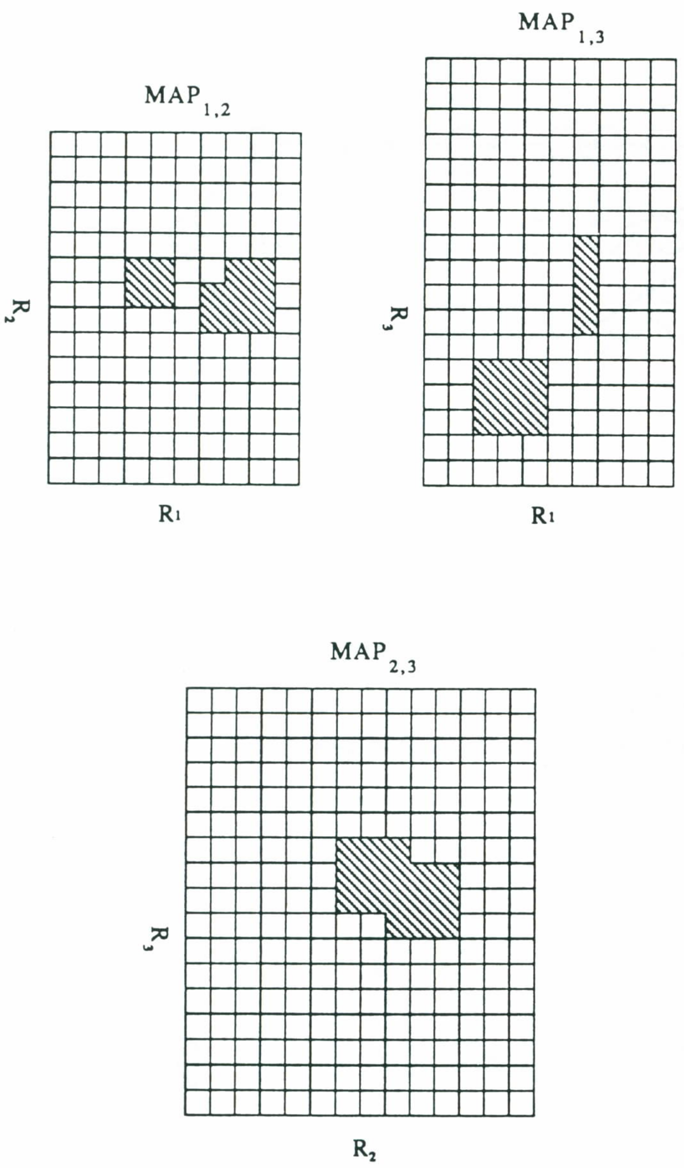


Fig. 13. The schedule maps of R_1, R_2, and R_3.

shown in Figure 13 where the collision regions are shown as shaded. Various operation modes of the three robots are scheduled using the improved scheduling method presented in the last section. The initial and optimum graphs for operation modes 1 and 2 are shown in Figure 14 where the critical paths are marked as bold lines. The final results associated with the Gantt charts for different robot operation modes (mode 1, mode 2, and mode 3 with and without priority assignment) are depicted in Figure 15 through Figure 18. Finally, a comparison table to show the effectiveness of the removal of redundant arcs in the improved method is given in Table I. It can be easily seen from Table I that the number of nodes generated in the search tree for each robot operation mode is highly reduced, so the efficiency is greatly increased.

6. CONCLUSIONS AND DISCUSSIONS

A new approach to motion planning for multiple robots with various operation modes is presented in this study. An advantage of the approach over the other previous work is that it can simultaneously generate safe motion for multiple robots. The concept of schedule map introduced in ref. 11 is employed. A map is created for each pair of the robots. By the assistance of the maps, the problem can be reduced to that of finding a minimaximal path in a disjunctive graph which can then be solved by an extension of the Balas algorithm. Additional constraints are enforced for various robot operation modes, and the scheduling results with minimum completion time are obtained after planning. The heuristic rules for defining the initial graph for tree search and the techniques for reducing the overall processing time are also presented.

The accuracy of the planning results can be improved by increasing the resolution of the schedule map, i.e. by increasing the number of the time intervals for each robot to complete its own task, or just by reducing the

Table I. A comparison table

Operation mode	Number of nodes generated	Value of best obtained (h)	Processing time† for detecting redundant squares	Total processing time†
Mode 1	103	19	—	11.70 secs.
Mode 1*	5	19	0.05 sec.	0.17 sec.
Mode 2	23	21	—	1.43 secs.
Mode 2*	2	21	0.05 sec.	0.11 sec.
Mode 3 (non-priority)	4,419	22	—	2.50 hrs.
Mode 3* (non-priority)	19	22	0.06 sec.	0.88 sec.
Mode 3 (with-priority)	608	28	—	375.69 secs.
Mode 3* (with-priority)	18	28	0.06 sec.	0.66 sec.

* By using the improved planning method.
† CPU time on PC-386.

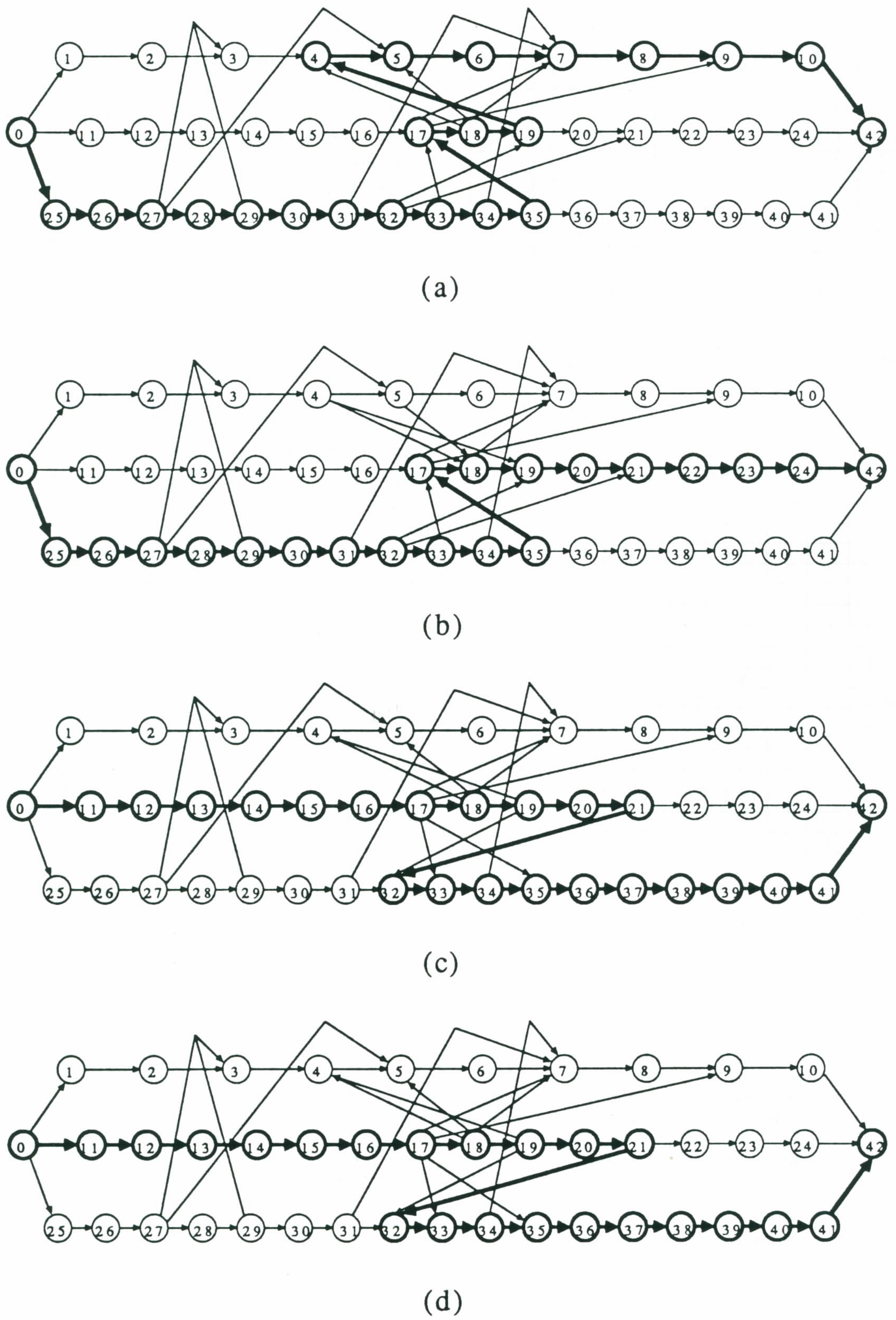


Fig. 14. The graphs corresponding to the root node and the optimal node of the search tree. (a) The initial graph G_0 of mode 1. (b) The optimal graph G^* of mode 1. (c) The initial graph G_0 of mode 2. (d) The optimal graph G^* of mode 2.

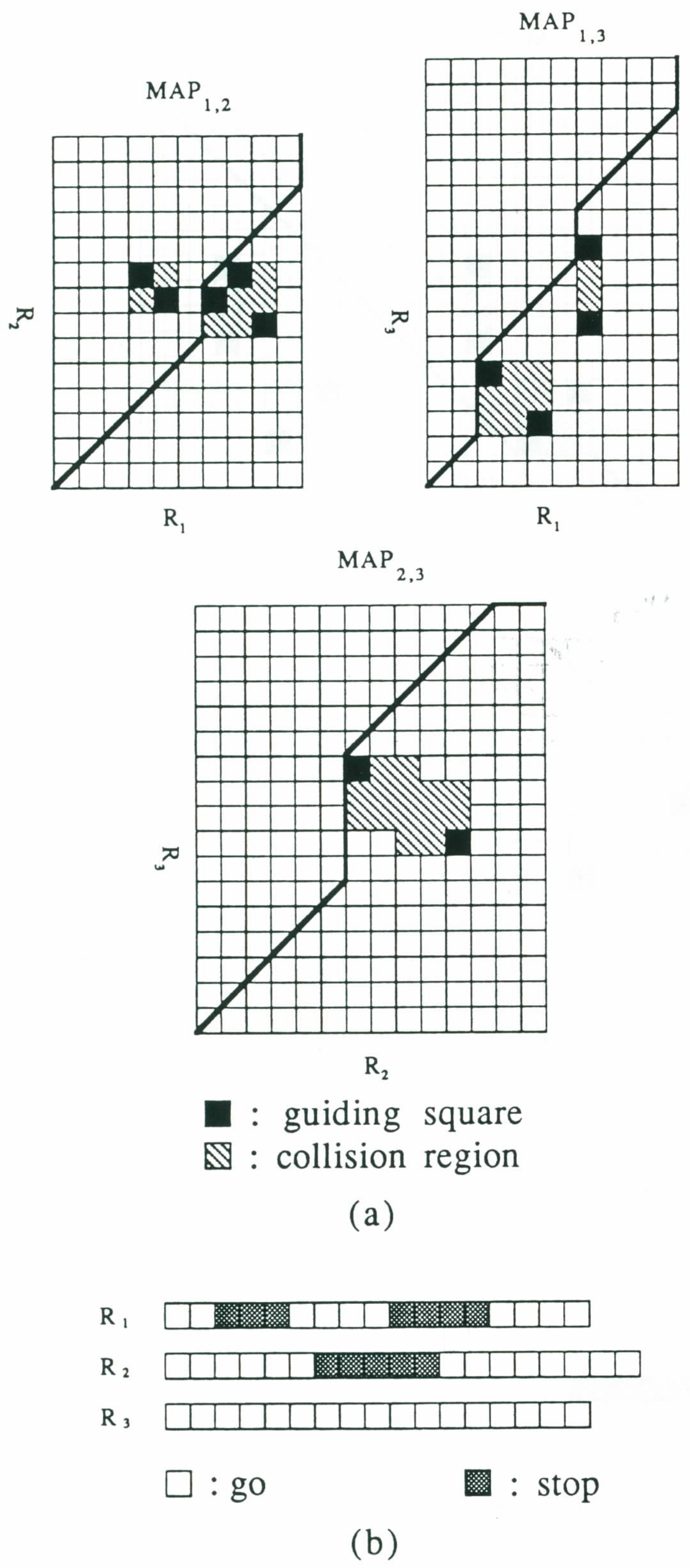


Fig. 15. The scheduling result of mode 1. (a) The schedule lines. (b) The corresponding Gantt charts.

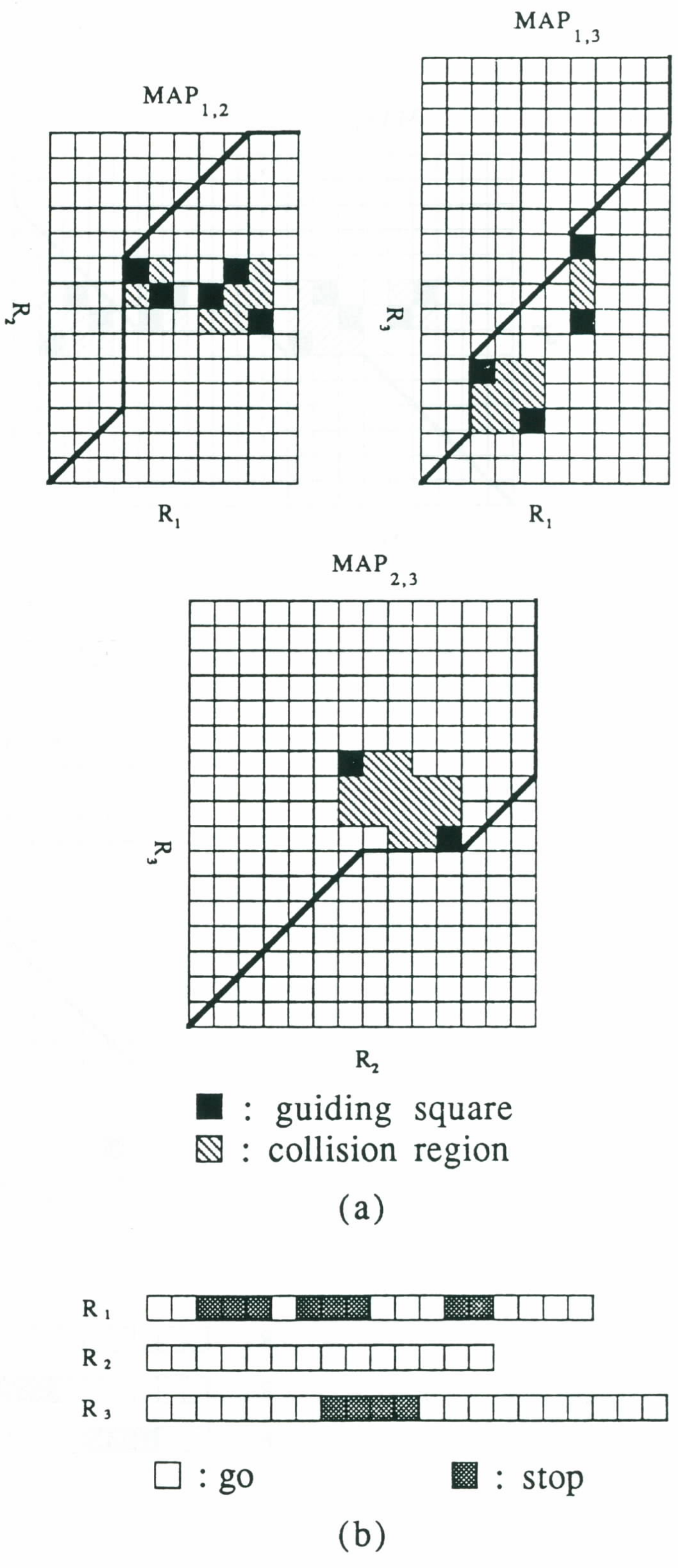


Fig. 16. The scheduling result of mode 2. (a) The schedule lines. (b) The corresponding Gantt charts.

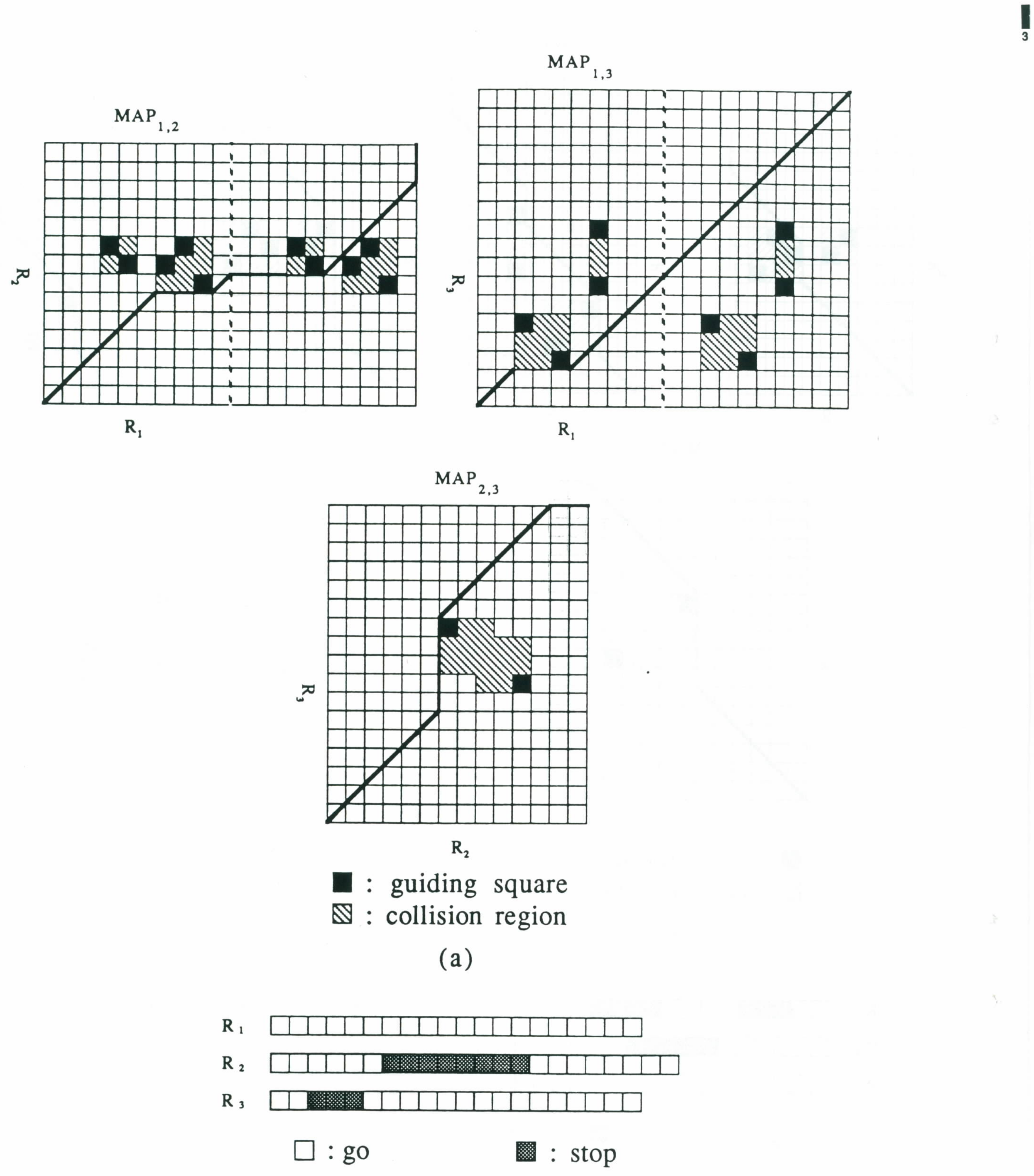


Fig. 17. The scheduling result of mode 3 without priority assignment. (a) The schedule lines. (b) The corresponding Gantt charts.

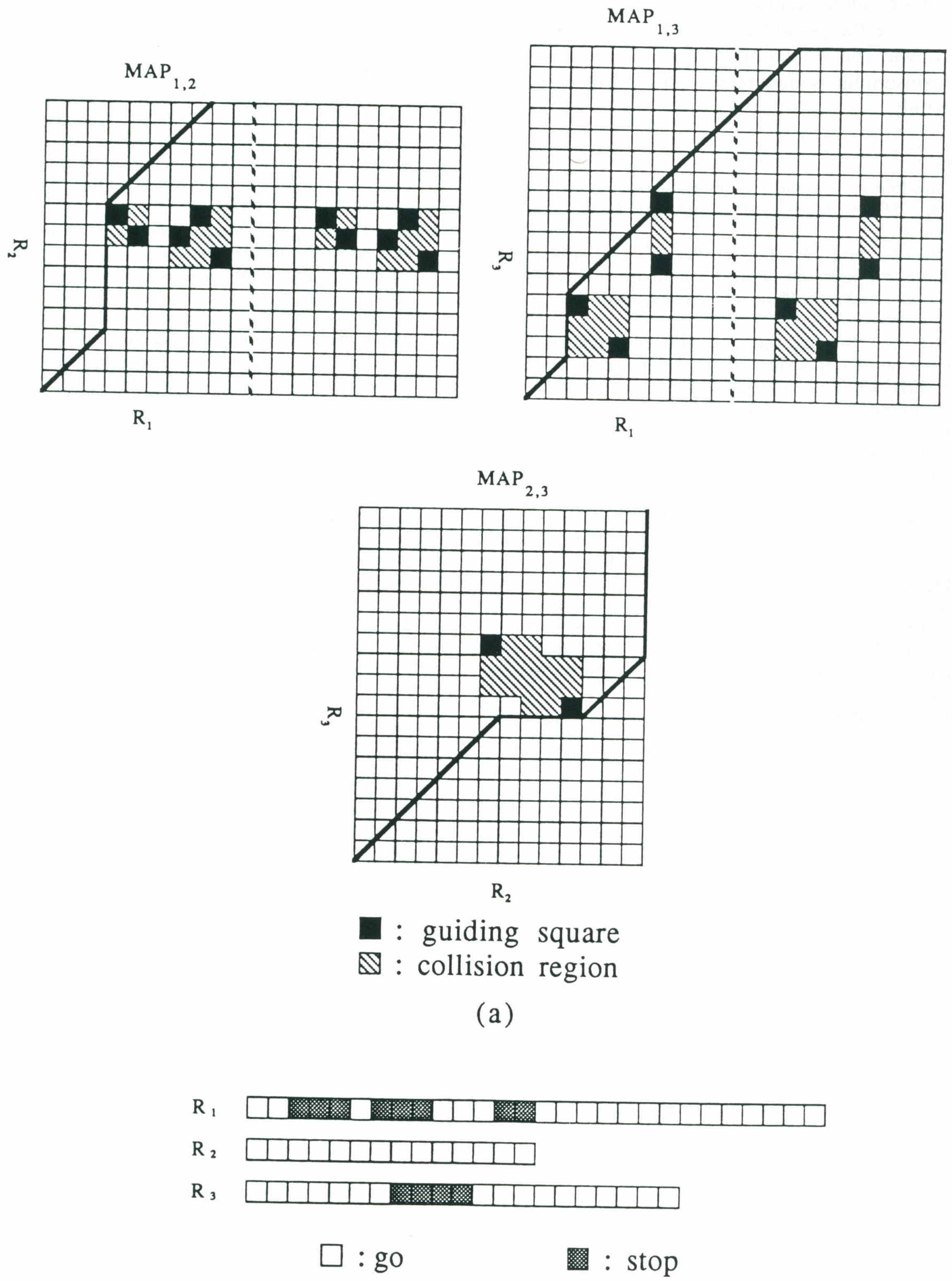


Fig. 18. The scheduling result of mode 3 with priority assignment. (a) The schedule lines. (b) The corresponding Gantt charts.

length of the time interval (i.e. the constant h). But this also increases the processing time required for the construction of the maps.

The planner implemented in this study cannot dynamically solve the planning problem; when one robot is broken down, the re-planning process which may be very time consuming is inevitable. This problem is left for future study.

References

1. T. Lozano-Pérez, "Spatial planning: a configuration space approach" *IEEE Trans. on Computers* **C-32,** 108–120 (Feb., 1983).
2. R.A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation" *IEEE Trans. on Syst., Man, and Cybern.* **SMC-15,** No. 2, 224–233 (1985).
3. R.A. Brooks, "Solving the find-path problem by good representation of free space" *IEEE Trans. on Syst., Man, Cybern.* **SMC-13,** No. 3, 190–197 (1983).
4. T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators" *IEEE J. Robotics and Automation* **RA-3,** 224–238 (June, 1987).
5. R. Ruff and N. Ahuja, "Path planning in a three dimensional environment" *Proc. Int. Conf. on Pattern Recognition,* Montreal, Canada 188–191 (1984).
6. B.H. Lee and C.S.G. Lee, "Collision-free motion planning of two robots" *IEEE Trans. on Syst., Man, Cybern.* **SMC-17,** No. 1, 21–32 (1987).
7. Y.H. Liu et al., "A practial algorithm for planning collision-free coordinated motion of multiple mobile robots" **In**: *Proc. IEEE Int. Conf. on Robotics and Automation,* Washington, D.C. (IEEE Computer Society Press, New York, 1989) pp. 1427–1432.
8. K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles, *IEEE Trans. on Robotics and Automation* **5,** 61–69 (Feb., 1989).
9. K. Kant and S.W. Zucker, "Toward efficient trajectory planning: the path-velocity decomposition" *Int. J. Robotics Research* **5,** 72–89 (Fall, 1986).
10. M. Erdmann and T. Lozano-Pérez, "On multiple moving objects" *Algorithmica* **2,** No. 4, 477–521 (1987).
11. C.F. Lin and W.H. Tsai, "Trajectory modeling, collision detection, and motion planning for robot manipulators" *NSC Technical Report No. NSC78-0404-E009-04, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China (submitted for publication).*
12. E. Balas, "Machine sequencing via disjunctive graphs: an implicit enumeration algorithm" *Operations Research* **17,** No. 6, 941–957 (1969).
13. S. Ashour, T.E. Moore and K.Y. Chiu, "An implicit enumeration algorithm for the nonpreemptive shop scheduling problem" *AIIE Transactions* **6,** No. 1, 62–72 (1974).
14. M. Florian, P. Trepant and G. McMahon, "An implicit enumeration algorithm for the machine sequencing problem" *Management Science* **17,** B782–B792 (Aug., 1971).
15. H.H. Greenberg, "A branch-bound solution to the general scheduling problem" *Operations Research* **16,** No. 2, 353–361 (1968).
16. W.W. Hardgrave and G.L. Nemhauser, "A geometric model and a graphical algorithm for a scheduling problem" *Operations Research* **11,** No. 6, 889–900 (1963).
17. B.J. Lageweg, J.K. Lenstra and A.H.G.R. Kan, "Job-shop scheduling by implicit enumeration" *Management Science* **24,** 441–450 (Dec., 1977).
18. J.K. Lenstra et al., "Complexity of machine scheduling problems" *Ann. Discrete Math.* **7,** 343–362 (1977).
19. M.R. Garey, D.S. Johnson and R. Sethi, "The complexity of flowshop and jobshop scheduling" *Mathematics of Operations Research* **1,** No. 2, 117–129 (1976).
20. J.W. Boyse, "Interference detection among solids and surfaces" *Comm. of the ACM* **22,** 3–9 (Jan., 1979).