Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

Parallelizing Planning and Action of a Mobile Robot Based on Planning-Action Consistency

Jun Miura and Yoshiaki Shirai
Dept. of Computer-Controlled Mechanical Systems,
Osaka University, Suita, Osaka 565-0871, Japan
jun@mech.eng.osaka-u.ac.jp

Abstract

This paper proposes a novel method to schedule parallel execution of planning and action of a mobile robot. The method considers the following two types of parallelism. (1) acting while planning*: if a partial planning result can be used to determine feasible actions, such actions can be executed while the planning process is still going. (2)* planning while acting*: if the result of the current action is (at least partially) predictable, the planning for the next action can start in advance of the completion of the current action. The proposed method uses the notion of* planning-action consistency *to guide the scheduling. The method has been successfully applied to a mobile robot navigation problem under sensor uncertainty.*

1 Introduction

Resource limitation and uncertainty are two important issues in planning for an agent in the real world. These two issues are closely related to each other; since planning under uncertainty is usually costly, the limitation of computational resources tends to be critical. Controlling the planning process by explicitly considering the planning cost is one approach to improvement of the overall efficiency [1]. Another approach is to schedule parallel execution of an agent's several activities such as reasoning, sensing, and action. This paper studies the parallel scheduling in a mobile robot navigation problem in which planning and vision-motion operations can run in parallel.

Fig. 1 shows an example problem treated in this paper. A mobile robot, which has a rough map of the environment, is going to the destination while selecting routes. There is a route which passes the narrow space (called *gate*); however the passability of the gate is initially unknown due to the uncertainty of visual data obtained at the initial position. The detour passing through the hallway is known to be passable, although it is longer. The robot estimates the gate width with stereo vision to determine the passability. The objective of planning is to generate a sequence of observation points which leads the robot to the destination efficiently. Such a situation is quite usual; for example, in a typical office environment, the position of desks, chairs, and other furniture are roughly known, while their exact locations are uncertain; some chairs may block the robot from taking a certain path to the destination.

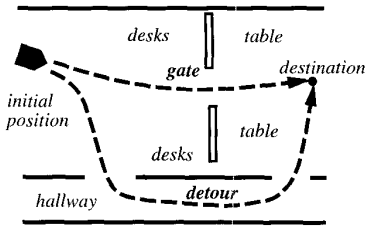


Fig. 1: An example problem.

This navigation problem can be hierarchically decomposed into the following two subproblems:

- The high-level *planner* generates a *subgoal* (i.e., an observation point).
- The low-level *action controller* determines an action (i.e., movement and observation) to achieve the subgoal and executes it.

This paper investigates the parallel scheduling of planning and action in this hierarchical structure.

2 Two Types of Planning-Action Parallelism

This section considers the following two types of planning-action parallelism: *acting while planning* and *planning while acting*. First, suppose the case where planning and action are sequentially performed. In this case, the following two steps are repeated: an action is determined by its preceding planning; a planning is performed based on the result of its preceding action. For parallelizing planning and action, therefore, one activity (planning or action) has to predict the result of the other activity in order to start before the completion of the other. The above-mentioned two types of parallelization are possible depending on which activity's result is predicted.

2.1 Acting while Planning

The *acting while planning* parallelism is to predict the result of the current planning and to start an action before the current planning finishes (see Fig. 2(a)). This type of parallelism has little been considered so far, but is important to improve the overall performance in the case where planning requires much computation due to uncertainty of information.

0-7803-6475-9/01/$10.00© 2001 IEEE

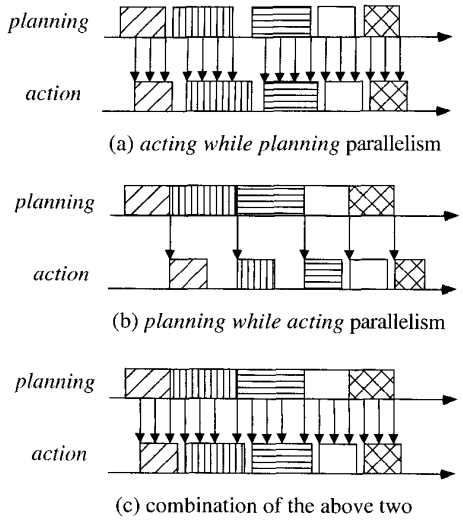


Fig. 2: Two types of parallelism and their combination.

2.2 Planning while Acting

The *planning while acting* parallelism is to predict the result of the current action and to start the planning for the next action before the current action finishes. Realtime search (e.g., [2]) or interleaving (e.g., [3]) is a suitable platform for this type of parallelism; that is, during execution of a selected action, the planner can search for the next action at the new state to be achieved by the current action. This parallelism is illustrated in Fig. 2(b). The planner sends a selected action (or action sequence) to the action controller for execution. Once the action is completed, the action controller waits for the next selected action.

Horvitz [4] proposed the models of *continual computation* which consider the effective use of the idle time between occurrences of problem instances. He showed several criteria and methods to determine which problem instances should be examined proactively based on the knowledge of forthcoming problem instances, such as a probabilistic distribution of their occurrences.

3 Parallel Scheduling Based on Planning-Action Consistency

This section describes our strategy for realizing both *acting while planning* and *planning while acting* parallelisms (see Fig. 2(c)). For scheduling, we need to decide a pair of parallelizable planning and action. For this purpose, we propose to introduce the notion of *planning-action consistency*, which states what actions are consistent with the current action and vice versa. A concrete planning and scheduling algorithm is described later.

3.1 Strategy for Acting while Planning

In the hierarchical decomposition of problems described in Sec. 1, the high-level planning process of subgoal determination can be viewed as a process of gradually reducing the subgoal candidates towards the final commitment to one subgoal [5]. If there are some ground-level actions which are *consistent* with (or, do not largely conflict with) the remaining subgoal candidates, such actions can be executed while the high-level planning process is still going.

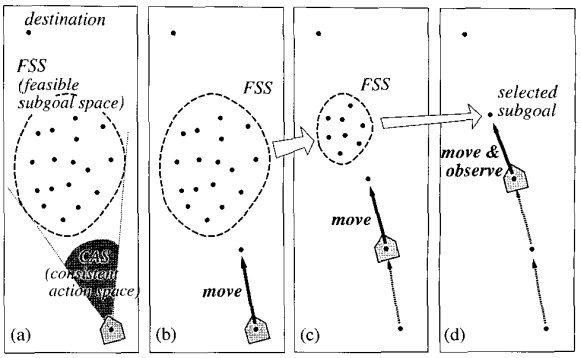


Fig. 3: Illustrative example of *acting while planning*.

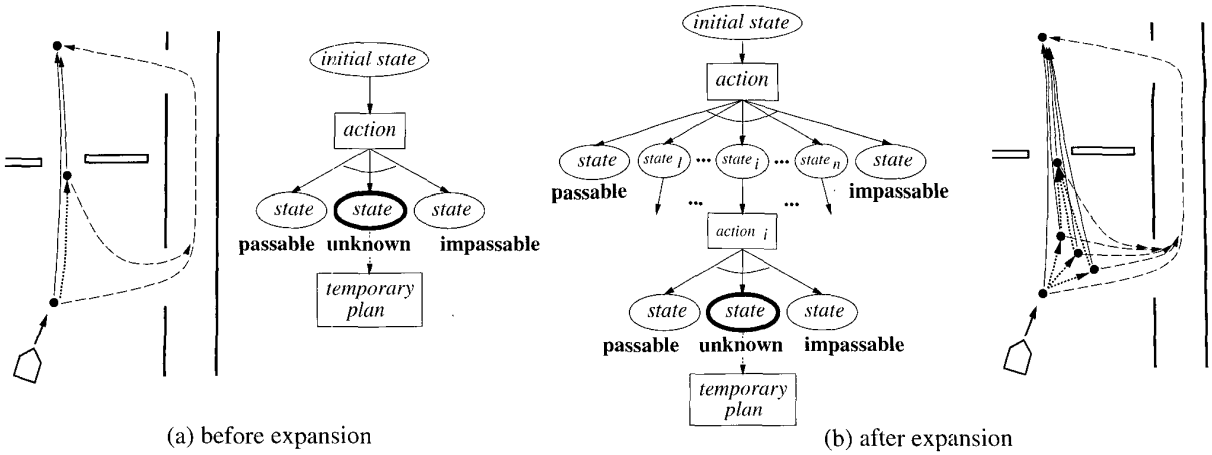
Fig. 3 illustrates the above idea. First, the initial set of feasible subgoal candidates is derived, which are collectively called an *FSS (feasible subgoal space)* (see Fig. 3(a)). Then, a set of actions which are consistent with the FSS, called *CAS (consistent action space)*, is selected (e.g., the shaded area in Fig. 3(a)). Among the candidate actions, the best one is selected and executed (see Fig. 3(b)). As the planning process continues, the FSS is reduced and actions are selected and executed repeatedly (see Fig. 3(c)). Finally one subgoal is selected and, a few moments later, the robot reaches the subgoal (observation point) and makes an observation (see Fig. 3(d)). If planning and action are sequentially performed, the robot is still at the initial position when the commitment to the subgoal is made. Thus, the distance which the robot travels until the commitment is made is the merit gained by the *acting while planning* parallelism.

Since the planning and the action processes can be executed asynchronously and in parallel, the space of possible schedules could be too huge to search. Therefore, we limit the timing of changing actions only to the end of each candidate-reducing cycle of planning.

To implement the above *acting while planning* scheduling, we need the following:

- A planning process which iteratively reduces the candidates (*iterative refinement planning*),
- A criterion to evaluate the consistency between an action and an FSS (*consistency criterion*), *and*
- A criterion to stop the iteration and to commit to one subgoal (*commitment criterion*).

The above two criteria are problem-specific; the concrete criteria for our planning problem is described in Sec. 5.1.



(a) before expansion (b) after expansion

Fig. 4: Expansion of an open node of a plan candidate and its corresponding situation. Ellipses drawn with bold lines indicate open nodes. In the leftmost and the rightmost figures, solid arrows, dashed arrows, and dotted arrows correspond to movement when the gate's state is passable, impassable, and unknown, respectively.

3.2 Strategy for Planning while Acting

We consider the effective use of the time between the commitment to one subgoal and its completion (see Fig. 3(d)). Suppose we can predict and enumerate a set of possible states at the completion of the current action, probably using a probabilistic action model. To proactively plan the next action for one predicted state, which we call a *planning option*, can be considered to be *consistent* with the current action. A set of such planning options is performed in parallel with the current action.

If the robot has enough time for this proactive planning, all planning options are performed. If not, a set of planning options is selected for execution based on the expected utility of planning. Since the allocated time for proactive planning is predictable, we apply optimization techniques such as linear programming to this selection problem. *Contract anytime algorithms* [6] may also be suitable for such optimization.

4 Iterative Refinement Planning

4.1 Plan Representation

A state is represented by the estimate of the gate width and the robot position. Due to vision uncertainty, the robot cannot measure the exact gate width but obtains its probabilistic distribution. After an observation, the robot classifies the state of the gate into one of the three categories (*passable*, *impassable*, and *unknown*) according to the relationship between the probabilistic distribution and the robot width [7].

Since the actual state after an observation depends on the observation result and cannot be determined beforehand, a subplan is generated for each possible state. Such a *contingent* plan is represented by an AND/OR tree as shown in Fig. 4. A node for an unknown state is called *open*. The quality of a plan is measured in terms of its execution cost, which is the *expectation* of the total execution time for movement *and* observation.

4.2 Iterative Refinement Formulation [8]

We formulate the planning process as an anytime iterative refinement process [9]; namely, the planner searches the space of feasible plans (executable plans) for the final plan. This formulation entails an *easily-obtainable* feasible plan for any open node. We use the following one:

> *The robot moves from the current position to the position just before the gate*[1] *. If the gate is passable, the robot passes it; if not, the robot takes the detour from that position.*

We make every plan candidate feasible by *temporarily* assigning this feasible plan to all of its open nodes and calculate its *temporary cost*.

A plan candidate is refined as follows. Before expansion, an unknown state is treated as one open node and has a feasible plan with it (see Fig. 4(a)). The expansion of the open node consists of discretizing it with some granularity, searching for the best action for each discretized state, and assigning the feasible plan to newly generated open nodes (see Fig. 4(b)). Note that the plan shown in Fig. 4(a) is one of the initial feasible plans. Also note that, at present, the granularity for discretization is predetermined to be a constant value.

One iteration consists of the following steps:

[1] At this position, the robot is assumed to be able to measure the gate width without uncertainty.

1. Refine all currently feasible candidates and calculate their temporary costs. Let C_p^{temp} be the temporary cost of plan candidate p.
2. Calculate the cost C_{fp^*} of the *incumbent* fp^* (the best feasible plan among those which have been obtained so far) as the minimum of C_p^{temp}'s.
3. Predict the new cost C_p^{new} of each candidate p after the next refinement step. Suppose we can predict the plan improvement (i.e., cost reduction) ΔC_p, which will be obtained by expanding all of its open nodes[2] . Then, C_p^{new} is given by $C_p^{temp} - \Delta C_p$. Remove candidates whose new cost is larger than C_{fp^*} and goto 1 with the reduced candidate set.

When there is no candidate whose new cost is less than C_{fp^*} in step 3, the planner terminates the iteration and commits to fp^*. The other termination condition, which is the commitment criterion, is described in Sec. 5.1.2.

Note that although the planning process iteratively refines plan candidates, the refinement at deeper levels than the first level is required only for comparing competing candidates.

5 Scheduling Algorithm

The proposed method performs *acting while planning* and *planning while acting* alternately. Both types of scheduling are performed so that the expected time to reach the destination is minimized. Fig. 5 shows a typical time chart of parallel scheduling. Note that the initial FSS is not calculated if it has been completed by the preceding *planning while acting*.

5.1 Scheduling for Acting while Planning

5.1.1 Feasible Subgoal Space and Consistent Actions

Feasible plan candidates are the ones whose predicted new costs are less than the cost of the incumbent (see Sec. 4.2). The current feasible subgoal space (FSS) is thus constructed as the set of the first subgoals (i.e., observation points) of the feasible plan candidates.

Next we define the consistency criterion. Basically we consider that a consistent action is the action which reduces the distance to *every* subgoal candidate. By additionally

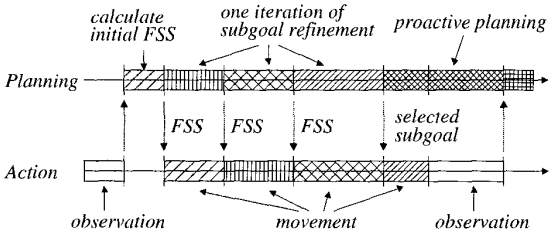


Fig. 5: A time chart of parallel scheduling.

[2] An example method to predict plan improvement is proposed in [8].

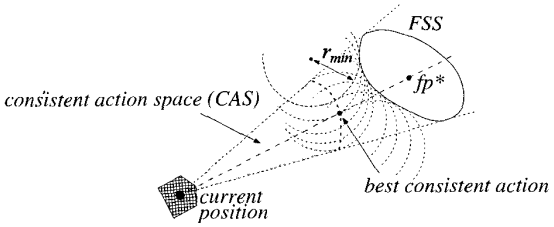


Fig. 6: Consistent action space and best consistent action.

using a problem specific knowledge that a winding movement of the robot is not preferable, we define the consistency criterion as follows (see Fig. 6).

We first set the search area for consistent actions surrounded by the two tangent lines connecting the current position and the two outmost position of the FSS. In this area, we collect points from which the robot can reach any candidate by a single circular trajectory satisfying the following two conditions:

- The radius of the trajectory is no less than the robot's minimum turning radius r_{min}.
- The robot does not turn more than 90 degrees on the trajectory.

The collected points constitute the consistent action space (CAS); that is, the movement to such a point is consistent with the current FSS. The above criterion is also applied to the cases where an FSS is composed of multiple clusters of subgoal candidates in the 2-D space.

If there is no consistent action due to a large expanse of the FSS, the robot performs the next refinement without motion. A typical example of such a situation is the case where the FSS is composed of two clusters, one at the north and the other at the south of the robot.

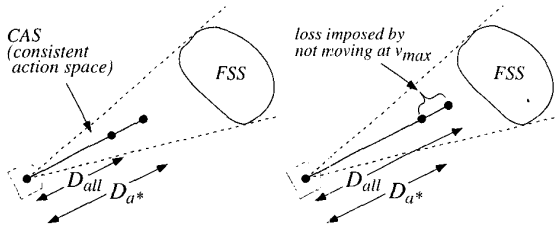
5.1.2 Determining Best Action

We choose an action which directs to the currently best feasible plan fp^*. In *acting while planning*, the robot maximizes the moving distance to minimize the execution cost, as long as it remains inside the CAS. Therefore, one candidate action is to move to the CAS boundary on the line directing to fp^* (see Fig. 6).

To determine the action during planning, we consider the following two distances. D_{a^*} is the moving distance of the above action; D_{all} is the distance covered by the robot moving at its maximum speed v_{max} for the duration of the next refinement step.

We estimate the time to perform the next refinement step as follows. For each candidate p, we calculate the cost (i.e., time) of expansion, which is the sum of expansion costs for p's open nodes[3] . The time for the next refinement is then calculated as the sum of all such costs.

[3] The expansion cost of an open node is basically determined from the number of candidate actions to be examined.



(a) $D_{all} \leq D_{a^*}$ (b) $D_{all} > D_{a^*}$

Fig. 7: Two possible relationship between D_{all} and D_{a^*}.

The relationship between D_{all} and D_{a^*} has an important role in action selection. Possible cases are enumerated as follows.

(Case 1): If $D_{all} \leq D_{a^*}$ (see Fig. 7(a)), since the robot can finish the next refinement step before exiting from the CAS, the robot can move at v_{max} while performing the refinement. After finishing the refinement, a new action is selected based on the updated FSS.

(Case 2): If $D_{all} > D_{a^*}$ (see Fig. 7(b)), performing the next refinement step instead of executing the current incumbent (fp^*) has a loss; namely, to perform the next refinement step, the robot has to move at a slower speed than v_{max} so that the robot stays inside the CAS; on the other hand, the robot can move at v_{max} in executing fp^*. The loss of the former behavior is calculated as the time needed to move the distance $D_{all} - D_{a^*}$ at v_{max}. Thus, we compare (i) C_{fp^*} and (ii) the minimum of the new cost (i.e., $\min_p C_p^{new}$) plus the loss, in order to select the behavior as follows.

(Case 2-a): If $C_{p^*}^{new} + loss \leq C_{fp^*}$, the robot moves by the distance D_{a^*} while performing the refinement. After finishing the refinement, a new action is selected based on the updated FSS.

(Case 2-b): If $C_{p^*}^{new} + loss > C_{fp^*}$, the robot stops the iterative refinement process and executes fp^*. This is *the* commitment criterion. While executing this fp^*, the robot performs a proactive planning based on the *planning while acting* scheduling.

5.2 Scheduling for Planning while Acting

5.2.1 Consistent Planning Options

We consider that a planning option is consistent with the current action if it has a possibility of being executed after the completion of the action. Since further planning is necessary in the cases where the gate's passability is unknown after observation, a planning option for a predicted *unknown* state is consistent. Note that unknown states can be enumerated because we use a predetermined granularity for discretization.

5.2.2 Determining Optimal Set of Planning Options

A planning option is one refinement step for an unknown state. We select a set of planning options so that the expected utility is maximized within the allocated time for *planning while acting*, T_{pwa}. T_{pwa} is calculated as the time between commitment to one subgoal and its completion. The utility is measured by the time saved by proactively executing planning options; thus the time for executing each option itself is the utility.

We first generate the initial FSS for each possible *unknown* state to estimate the expected execution cost of each option. For the ith state, let P_i be its probability and t_i be its execution cost of the next refinement. The expected utility of executing the ith planning option is given by $P_i t_i$. Let T'_{pwa} be the remaining time, given by subtracting the time for the initial candidate generation from T_{pwa}. If the total execution cost (i.e., $\sum_{i=1}^{n} t_i$) is less than T'_{pwa}, then all options are executed. If not, we solve the following optimization problem:

- objective: $\sum_{i=1}^{n} P_i t_i w_i \quad \longrightarrow$ maximum,
- constraints: $\sum_{i=1}^{n} t_i w_i \leq T'_{pwa}, \quad w_i = 0$ or 1,

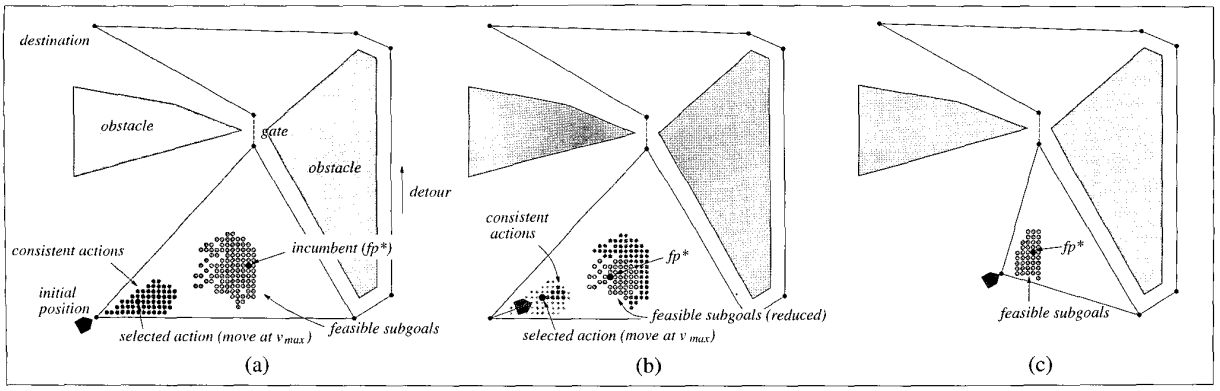
where w_i is the variable which indicates whether the ith planning option is executed. This is a 0-1 knapsack problem [10].

If there still remains some amount of time after executing all planning options, we proceed to the deeper levels of refinement until the time is exhausted.

6 Simulation Results

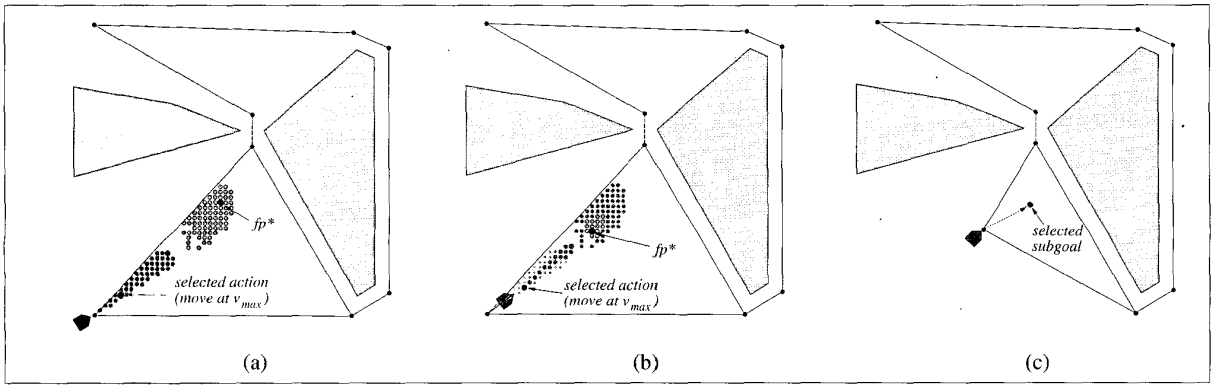
Figs. 8 and 9 show the planning problem used for simulation and the two results with different initial probabilities of the gate being passable. The robot moves directly to the gate if it is passable; otherwise, the robot moves directly to the entrance to the detour. Since the possible position of the next observation point (subgoal) is limited to the inside of the triangle shown in the center of the figure, we set grids for subgoals (observation point) and actions (goal points of actions) in this area. The other lines indicate possible paths of the robot from the observation area to the destination. The minimum path length from the initial position to the destination is about 680 $[cm]$; the maximum speed (v_{max}) is 5 $[cm/s]$; the time needed for one observation is 4 $[s]$. The granularity of discretization (see Sec. 4.2) is 5.

Fig. 8 is the result for the initial probability of 0.4. The time the robot spent until the passability was determined was 119.7 $[s]$. The time savings over the sequential method made by *acting while planning* and *planning while acting* were 17.23 $[s]$ (12.6%) and 3.55 $[s]$ (2.9%), respectively. Fig. 9 is the result of the initial probability of 0.7. The time until the passability was determined was 93.0 $[s]$; the time savings were 12.9% and 1.3%, respectively. Since the criterion for planning is to minimize the expected time to the destination, we expect that the next observation point will be near the left-upper edge of the triangle for a high initial probability and will be near the lower edge for a low initial probability. The FSSs in these two results appear as expected.



(a) From the initial FSS (the right cluster of points) and the CAS (the left one), the movement at v_{max} was selected and executed. During this movement, the FSS was updated. (b) From the updated FSS and the corresponding CAS, the movement at v_{max} was selected again. After the movement, the subgoal denoted as fp^* was selected. During executing fp^*, the proactive planning was performed for 7 out of 17 predicted *unknown* states. (c) The robot reached the subgoal and observed the gate. The actual state after the observation was the one for which the proactive planning had been performed. However, the reduced FSS generated by the proactive planning was still to large to obtain consistent actions; so the robot continued the refinement without motion and finally selected fp^* as the next subgoal.

Fig. 8: Simulation result 1.



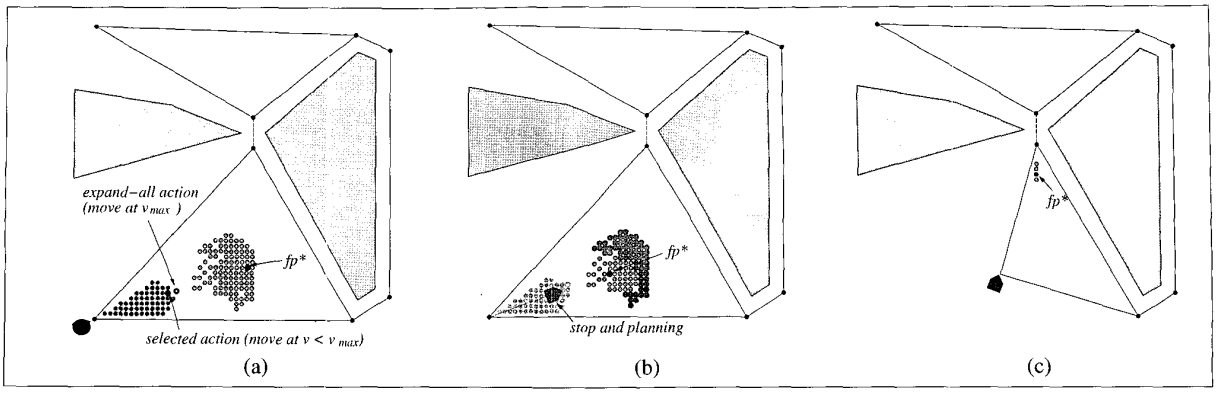
(a) From the initial FSS (the right cluster of points) and the corresponding CAS (the left one), the planner selected an action of moving at v_{max}. (b) After two cycles of refinement, fp^* was selected. During executing fp^*, all planning options were performed. (c) Since a next subgoal for every predicted state had been obtained, the robot immediately moved to the next subgoal.

Fig. 9: Simulation result 2.

Table 1 shows the effect of the initial probability on the averaged savings (of 100 trials) made by parallel scheduling. The savings by *planning while acting* are very low because only one of many examined states actually occurs; so we also show the maximum savings. Basically the saving made by parallel scheduling depends on the number of plan candidates examined; thus, the more uncertain the situation is (i.e., the initial probability is medium), the more effective the parallel scheduling is. Table 1 roughly shows such a tendency.

Table 1: Average savings made by parallel scheduling. (awp:*acting while planning*, pwa:*planning while acting*)

initial prob.	by awp (%)	by pwa (%)	(maximum)
0.1	13.46	0.01	(1.07)
0.3	14.81	0.16	(3.00)
0.5	16.01	0.16	(3.29)
0.7	16.48	0.20	(3.04)
0.9	10.87	0.15	(2.58)



(a) Since the condition **(Case 2-a)** in Sec. 5.1.2 was satisfied, an action of moving at the speed of 92% of v_{max} was selected. (b) Since no CAS was obtained, the robot performed refinement while stopping and selected fp^*. (c) Based on the updated FSS, the robot moved at v_{max} with refinement and selected fp^*.

Fig. 10: Simulation result 3.

Fig. 10 is the result of simulation whose parameter setting is the same as Fig. 8 except the doubled robot maximum speed (i.e., $v_{max} = 10\,[cm/s]$). Since the robot had to slow down from v_{max} to stay inside the CAS, the robot compared the following two actions (see Sec. 5.1.2): one is move at v_{max} with commitment to the current best feasible solution; the other is move at a slower speed than v_{max} while continuing the refinement. The robot finally selected the latter action.

7 Conclusion

We have proposed a novel parallel scheduling method which realizes not only *planning while acting* but also *acting while planning* parallelism. The notion of *planning-action consistency* is used for determining a candidate set of one activity which can be executed in parallel with the other activity. We detailed the scheduling algorithm for a mobile robot navigation problem under vision uncertainty. Simulation results show that the proposed method is promising. We are now planning to test the method on our real mobile robot.

The proposed scheduling method can basically be applied to other problems by considering the following. Concerning *acting while planning*, we need to devise an iterative refinement planner and to determine two criteria (*consistency criterion* and *commitment criterion*). It seems especially important to use a proper commitment criterion, because it could have a large effect on the degree of parallelism. Concerning *planning while acting*, we need to be able to predict the result of actions, at least probabilistically.

References

[1] S. Russell and E. Wefald. *Do The Right Thing*. The MIT Press, 1991.

[2] R.E. Korf. Real-Time Heuristic Search. *Artificial Intelligence*, Vol. 42, pp. 189–211, 1990.

[3] I. Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, 1997.

[4] E. Horvitz. Models of Continual Computation. In *Proceedings of AAAI-97*, 1997.

[5] S. Kambhampati, C.A. Knoblock, and Q. Yang. Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning. *Artificial Intelligence*, Vol. 76, pp. 167–238, 1995.

[6] S. Zilberstein. *Operational Rationality through Compilation of Anytime Algorithm*. PhD thesis, University of California at Berkeley, 1993.

[7] J. Miura and Y. Shirai. Vision and Motion Planning for a Mobile Robot under Uncertainty. *Int. J. of Robotics Research*, Vol. 16, No. 6, pp. 806–825, 1997.

[8] J. Miura and Y. Shirai. Vision-Motion Planning for a Mobile Robot considering Vision Uncertainty and Planning Cost. In *Proceedings of IJCAI-97*, pp. 1194–1200, 1997.

[9] M. Boddy and T. Dean. Solving Time-Dependent Planning Problems. In *Proceedings of IJCAI-89*, pp. 979–984, 1989.

[10] T. Ibaraki. Enumerative Approaches to Combinatorial Optimization. *Annals of Operations Res.*, Vol. 10 and 11, , 1987.