Real-Time Planning by Interleaving Real-Time Search with Subgoaling

Shigeo Matsubara
NTT Communication Science Laboratories
2 Hikaridai, Seika-cho, Soraku-gun,
Kyoto 619-02, Japan
matubara@cslab.kecl.ntt.jp

Toru Ishida
Department of Information Science
Kyoto University
Yoshidahonmachi, Sakyo-ku,
Kyoto 606-01, Japan
ishida@kuis.kyoto-u.ac.jp

Abstract

Recently, real-time planning has been actively studied for solving problems in uncertain and dynamic environments. RTA* is a real-time search algorithm that can provide a computational basis for real-time planning. However, RTA* is not always efficient since obtaining effective heuristic functions is difficult when the problem becomes complicated. In order to keep the problem simple enough for efficient search, we propose an algorithm called RTSS, which incorporates the STRIPS subgoaling function into RTA*. This algorithm interleaves subgoaling and real-time search processes by evaluating the goal complexity and the ease of operator execution. An analysis using a simple model shows that the search cost can be significantly reduced by switching between subgoaling and real-time search. Furthermore, experiments on a robot task planning problem show that RTSS can attain the goal without performing many superfluous actions, while other algorithms often tend to perform a blind search that fails to attain the goal.

Introduction

Real-time problem solving studies are meeting today's needs for applying AI technology to real world problems. Many applications (e.g. robot task planning, process control, etc.) require real-time planning, i.e. to select and execute some action by a certain deadline before a complete plan is obtained. Real-time A*(RTA*) [Korf, 1990] is a real-time problem solving method that performs the best first search. This algorithm has the following properties: (1) it interleaves planning and execution and (2) it commits an action in constant time.

RTA* is effective if a heuristic function returns an accurate value. Such an effective heuristic function can be obtained for simple problems. On the other hand, for complicated problems it is very difficult to obtain effective heuristic functions. If a heuristic function returns an inaccurate value, the algorithm cannot distinguish promising states from non-promising states. Consequently, RTA* tends to perform superfluous actions before attaining the goal.

Subgoaling is an effective technique for attaining complicated goals. For example, STRIPS [Fikes and Nilsson, 1971] selects an operator that reduces differences between the initial state and the goal state. If the operator's precondition is not satisfied in the initial state, the precondition is treated as a new subgoal. This procedure is repeated until an operator can be executed. STRIPS can be regarded as a planner that interleaves planning and execution, since some action can usually be executed before a complete plan is obtained.

However, it is not always possible to set appropriate subgoals. For problems such as path planning, STRIPS cannot set an appropriate subgoal based on the differences between the goal and the initial state. In such cases, STRIPS often fails to select the most effective operator because the selection criterion based on the number of literals added/deleted is not powerful enough. Consequently, STRIPS tends to perform superfluous actions to attain the goal.

This paper proposes an algorithm called Real-Time Search with Subgoaling (RTSS), which incorporates the STRIPS subgoaling function into RTA*. This algorithm continues to make a subgoal until the subgoal becomes simple enough or a certain deadline comes. The algorithm utilizes two indexes: degree of goal complexity (DGC) and degree of execution ease (DEE). The threshold on DGC determines the switching point from subgoaling to real-time search. An analysis using a simple model shows that the RTSS performance can be improved by tuning this threshold value. Operator selection based on DEE prevents RTSS from continuing the planning process for a long time. The experimental result of a robot task planning problem shows that RTSS can attain the goal by fewer redundant actions even when RTA* or STRIPS alone falls into a blind search and fails to attain the goal.

Previous Works and Their Drawbacks for Interleaved Planning

RTA*

RTA* iterates planning and execution alternately. The outline of the RTA* algorithm is as follows.

1. Calculate all neighboring states of the current state.
2. Evaluate the cost to the goal state from these states by a heuristic function.
3. Move to the neighbor with the minimum expected cost.
4. Update the heuristic function value in the previous state.

RTA* commits an action in constant time. However, there is no guarantee of finding the optimal solution. RTA* is effective as long as the heuristic function returns a value accurate enough to distinguish between right and wrong directions to the goal. However, it is not easy to obtain an effective heuristic function for complicated problems.

For example, suppose that the task for a robot is to bring two boxes, A and B, to their respective goal positions. In the initial state, box A is closer to the robot than box B. First, suppose that we use (a)"the sum of the distances between the initial and the goal positions of boxes A, B" as the heuristic function. If the robot cannot move the box in the current position, the robot's action does not affect the heuristic value. Therefore, the robot will move randomly and perform superfluous actions before attaining the goal.

Next, suppose that we use the sum of (a) and (b)"the distance between the robot and the box closest to it" as the heuristic function. In this case, the robot can complete the task of bringing box A to the goal position without performing many superfluous actions. However, the robot cannot go to box B's initial position easily because the robot's movement in that direction increases the value of (b).

In this way, it is necessary to modify the heuristic function many times to obtain a satisfactory plan. As a problem becomes more complicated, it becomes more difficult to obtain a satisfactory heuristic function.

STRIPS

In the STRIPS representation, a world state is described by a conjunction of literals. Actions are described by *operators*. An operator consists of three components: precondition, delete-list, and add-list. If the precondition of an operator matches a current state description, that operator can be executed. When an operator is executed, literals in the delete-list are deleted from a state description, and literals in the add-list are added to the state description.

An outline of the interleaving version of the STRIPS algorithm is as follows[†1].

1. Select the most effective operator for reducing differences between the current state and the goal state.
2. If the operator's precondition is satisfied in the current state, execute it. The current state changes.
3. If not, make that precondition a new subgoal and call the STRIPS procedure recursively.

Continue the above procedure until the current state matches the goal state.

STRIPS is effective if an appropriate operator can be selected based on the differences between the initial and goal states. Such an operator can be selected in the following cases: there is a single operator that reduces the differences; there are few interactions in the order of the subgoal attainments and there is a single operator for each subgoal; there is an operator that reduces the differences to zero. However, these cases do not happen very often. In other cases, an appropriate operator is not always selected, because appropriateness cannot be distinguished by counting literals added or deleted. In order to select an appropriate operator, it needs to take account of the semantics of the literals. If an appropriate operator cannot be selected, STRIPS functions as a blind backward search. In that case, STRIPS cannot perform any action before calculating a complete plan. Consequently, real-time problem solving becomes impossible.

Real-Time Search with Subgoaling (RTSS)

In order to overcome the drawbacks of RTA* and STRIPS, we propose an algorithm called Real-Time Search with Subgoaling (RTSS) which incorporates STRIPS-like subgoaling into RTA*. This algorithm evaluates whether the goal is complicated or not. If the goal is simple enough to find a satisfactory operator with the heuristic function, the algorithm applies RTA* to the subproblem. If it is complicated, RTSS continues to make subgoals until a subgoal is simple enough. By doing this, the RTA* part of RTSS can avoid falling into a blind search since RTA* is given a simple goal, the heuristic function returns an accurate value. The STRIPS part of RTSS can also avoid falling into a blind search since the search space in STRIPS is reduced.

Degree of Goal Complexity (DGC) and Degree of Execution Ease (DEE)

RTSS utilizes the following two indexes.

- Degree of Goal Complexity (DGC)

[†1]Note that the original version of STRIPS does not execute any action before a complete plan is obtained.

Given the initial state and the goal state, this index represents whether the goal is simple or complicated for the heuristic function. This index determines whether RTSS should continue to make subgoals or carry out real-time search.

- Degree of Execution Ease (DEE)

  Given an operator, this index represents how early the operator can be executed. If inappropriate operator selection is repeated, RTSS cannot easily switch to real-time search. This leads to give a real-time search part of RTSS not enough simple subproblems. This drawback is reduced by the operator selection based on this index.

Algorithm

The algorithm of RTSS is as follows.

Subgoaling(I,G)

1. Return if there is no difference between the initial state I and the goal state G.
2. If DGC $<$ the threshold value t, call **Real-Time Search(I,G)**
3. If DGC $\geq$ the threshold value t,
   - 3.1 Select an operator.
     - (a) Calculate the difference between the goal state G and the initial state I ($D = G - I$). D is a set of literals included in G but not satisfied in I.
     - (b) Find operators whose add-list includes more than one element of D.
     - (c) Select the operator o with maximum DEE.
   - 3.2 Call **Subgoaling(I,P)** recursively, where P is the precondition of operator o.
   - 3.3 Execute o and calculate the state S after the execution.
   - 3.4 Call **Subgoaling(S,G)** recursively.

Real-Time Search(I,G)

1. Let x be the initial state I.
2. Return if the state x is equal to the goal state G.
3. Calculate the states x' after applying a feasible operator to x.
4. Calculate the heuristic value $h(x')$ of all x'. $h(x')$ is the heuristic cost to attain the goal.
5. Let the second minimum $(h(x') + d)$ be the new heuristic value of the state x, where d is cost for moving from x to x'.
6. Move to the state with minimum $h(x')$, and go to 2 with x' as x.

The above algorithm does not guarantee that the subgoaling process finishes in constant time. However, if only the depth of recursively calling **Subgoaling** is restricted, it is easily guaranteed that some action can be executed in constant time.

If the threshold t is 1 (where DGC ≥ 1), RTSS becomes identical to STRIPS. If the threshold t is large enough, RTSS becomes identical to RTA*.

RTSS assumes that a goal state is reachable from every state, i.e. there is no state on the solution path where RTSS cannot escape.

The real-time search part may be regarded as the reactive system. In that case, RTSS is viewed that a planner (STRIPS) improves the behavior of the reactive system (RTSS). The idea related to this can be found in [Drummond *et al.*, 1993] where planning and execution are not interleaved.

Evaluation of RTSS Performance using a Simple Model

RTSS performance was evaluated by using a simple model. This showed that there was an optimal threshold value on DGC for planning cost reduction.

First, we estimated the plan length obtained by RTA*. Suppose that the state space is represented by a tree. Then, there exists only one goal node, which is located at depth d in the tree. Additionally, the probability of moving in the correct search direction to the goal is represented as p. This value represents the effectiveness of the heuristic function. The goal node is reached when the difference between the number of moves to the right search direction and that to the wrong search direction becomes d. We represent the number of combinations of $d+i$ right moves and i wrong moves as n_i for reaching the goal node by $d+2i$ steps. n_i is given as follows.

$$n_0 = 1 \quad (i = 0)$$

$$n_1 = \binom{d}{1} \quad (i = 1)$$

$$n_i = \binom{d+2i-2}{i} - \sum_{k=0}^{i-2} \binom{2i-2-2k}{i-k} n_k \quad (i \geq 2)^{\dagger 2}$$

Therefore, the expected value of needed steps l from the root to the goal node is given as follows.

[†2] When we reach the goal node by $d+2i$ moves, these moves include $d+i$ right moves and i wrong moves. If the $(d+2i)$ th move is wrong, we do not reach the goal node by the $(d+2i)$ th move. If the $(d+2i-1)$ th move is wrong, we have already reached the goal node in $d+2i-2$ moves. Therefore, there is $\binom{d+2i-2}{i}$ ways for reaching the goal node by $d+2i$ moves. However, the number of these ways includes such ways that allow us to reach the goal node in less than $d+2i$ moves. For example, the number of ways needed to reach the goal node by the d th move in $\binom{d+2i-2}{i}$ is $n_0 \times \binom{2i-2}{i}$ (there is i wrong moves in the remaining $d+2i-2-d(=2i-2)$ moves.). From this discussion, the expression for n_i is derived.

$$E[l]_{(d,p)} = \sum_{i=0} n_i(d+2i)p^{d+i}(1-p)^i \dagger 3$$

Second, we estimated the subgoaling effect. The positive effect appears as to increase the probability p mentioned above. This happens when the two-boxes problem is divided into two one-box problems. However, this effect does not affect the plan length obtained by STRIPS[†4]. The negative effect is that the plan length tends to be increased. This happens in such cases as path planning. In this case, the subgoaling becomes almost the same as a blind backward search. In this evaluation, we treat STRIPS as a backward search moving in the correct search direction to the goal with a probability q, i.e. $E[l]_{(d,q)}$ is the expected value of needed steps l to the goal by STRIPS.

Third, we calculated the plan length obtained by RTSS. RTSS incorporates subgoaling into RTA*. Suppose that the threshold is set so that m subgoals are made all over the problem solving process, where each subproblem has the goal at depth $(d-m)/(m+1)$ and the probability p' when solved by RTA*[†5]. However, because of the subgoaling negative effect, the STRIPS subgoaling function eventually makes more subproblems than expected, i.e. $(E[l]_{(m,q)}+1)$ subproblems are generated[†6].

From the above discussion, the three algorithms' planning costs are given as follows, where u_R and u_S are costs for calculating one length plan by RTA* and STRIPS, respectively.

$$Cost_{RTA*} = E[l]_{(d,p)}u_R$$

$$Cost_{STRIPS} = E[l]_{(d,q)}u_S$$

$$Cost_{RTSS} = (E[l]_{(m,q)}+1)E[l]_{(\frac{d-m}{m+1},p')}u_R + E[l]_{(m,q)}u_S \dagger 7$$

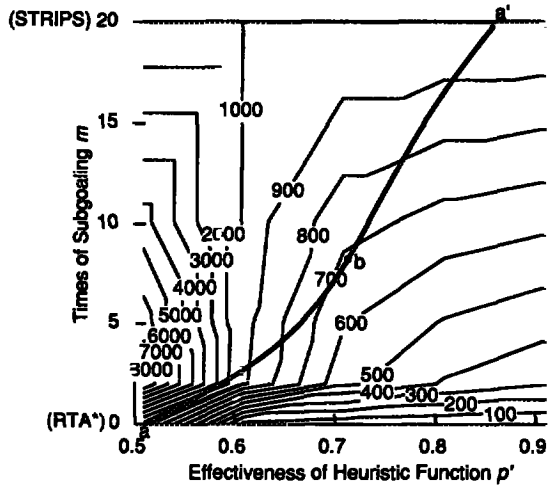

Figure 1: Planning Cost Contour Analyzed for a Simple Model ($d = 20, q = 0.55, u_R = 1, u_S = 5$)

Figure 1 shows the planning cost, given p' and m. When m is 0 or 20, the planning cost is identical to the cost of RTA* or STRIPS, respectively. The values of p' and m are interrelated and vary depending on the problem domain. For example, suppose that p' and m change along the curve between $a-a'$. As m increases, the planning cost first increases, next decreases, then increases again. The cost has its minimum value at point b. This result indicates that the cost can be reduced by appropriately tuning the threshold on DGC.

DGC and DEE based on Abstraction Level

In order to apply RTSS to a problem in practice, it is necessary to calculate DGC and DEE. In this paper, we use the following DGC and DEE based on the abstraction level [Sacerdoti, 1973]. Each literal belongs to one abstraction level. A higher abstraction level means that the literal represents a more abstract concept.

DGC

Suppose there are no literals belonging to the high abstraction level in the difference set D between the goal state G and the initial state I. In this case, a part of the goal belonging to the high abstraction level is already attained. Therefore, we can assume that RTA* is able to attain the remaining unsatisfied goal parts since the goal has already become simple. Accordingly, the heuristic function returns a highly accurate value in this situation.

DGC is defined as follows.

$$DGC(I,G) = \max_{d_i} f(d_i)$$

where $d_i \in D(= G - I)$, and $f(x)$ is the abstraction level of the literal x.

[†3] In RTA*, p is improved as the problem solving process progresses. However, this paper assumes that p is a fixed value (the average value over that process).

[†4] p is related to the RTA* performance. In general, subgoaling is not helpful in narrowing STRIPS's operator selection range.

[†5] We assume that the original problem is equally divided into subproblems. Prime(') is used to distinguish the probability in a subproblem from that in the original problem. Usually $p' \geq p$.

[†6] If there is no subgoaling negative effect, $m+1$ subproblems are generated. The m times subgoalings mean that STRIPS is given the problem whose shortest plan length is m. The subgoaling negative effect lengthen the plan length to $E[l]_{(m,q)}$.

[†7] In RTSS, real-time search and subgoaling are not carried out simultaneously. Therefore, the RTSS cost is calculated as the sum of each part cost. The first term is the cost by real-time search. Real-time search solves the problems with depth $(d-m)/(m+1)$, $(E[l]_{(m,q)}+1)$ times. The second term is the cost by subgoaling. Subgoaling is performed $E[l]_{(m,q)}$ times.

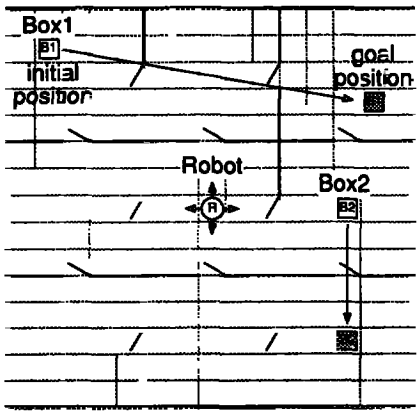

Figure 2: Robot Moving Space

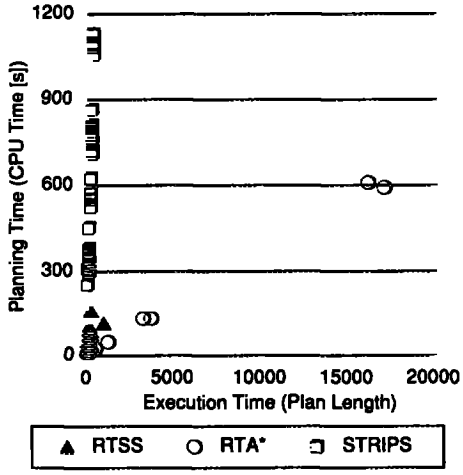

Figure 3: Experimental Results

DEE

First, those operators that assert subgoals are selected as candidates. We use the following strategy for selecting an operator among the candidates, i.e. the operator that asserts the literal in a high abstraction level first. By using this strategy, executable operators tend to appear earlier.

DEE is defined as follows.

$$DEE(Operator) = w\sum_{y_i}(f(y_i)\times 1) - (1-w)\sum_{z_i}(1/f(z_i))$$

where $y_i \in \{D \cap \text{Add-list}(Operator)\}$, $z_i \in \{G \cap \text{Delete-list}(Operator)\}$, $f(x)$ is the abstraction level of the literal x, and $w(0 \leq w \leq 1)$ is the weighting coefficient[†8].

Experiment

Problem for Evaluation

The performance of RTSS was evaluated in a robot task planning problem. This task involves bringing two boxes to their respective goal positions. A robot moves in the space shown in Figure 2. The whole space consists of nine rooms (3x3), each of which consists of twenty-five grids (5x5). The rooms are connected by doors. The robot is able to move forward, backwards, left or right by itself and to push one box. Additionally, the robot is able to open or close the door.

A state is described by the following six propositions: (**box-in** *box room*), (**robot-in** *room*), (**box-at** *box location*), (**robot-at** *location*), (**open** *door*), and (**close** *door*). These propositions are classified into two groups in terms of the abstraction level.

High(2): box-in, robot-in

Low(1): box-at, robot-at, open, close

The following six operators are used: **goto-room-location**, **push-box**, **go-through-door**, **push-through-door**, **open-door**, **close-door**.

An RTA* heuristic function is the sum of the Manhattan distances of the following:

- the distance between the boxes' initial positions and their respective goal positions
- the distance between the robot's current position and the box position nearest to the robot

For each algorithm, we solved thirty problems. In this evaluation, operator selection in STRIPS as well as in RTSS is based on DEE. In each problem, the initial positions of the robot and the boxes and their goal positions are set by random numbers. All doors are closed in the initial state.

The obtained plans are evaluated in terms of the planning time and the execution time. The former is measured by the required CPU time and the latter is measured by the length of the operator sequence in the plan. When the CPU time exceeds 1200 seconds for a trial, this trial is terminated and treated as a failure.

Evaluation Results

Figure 3 shows the performances of RTA*, STRIPS and RTSS (threshold $t = 3, 1, 2$, respectively). Each point in the figure represents the planning and the execution time to one trial. The evaluation provided the following observations.

- Planning time is proportional to execution time, since the size of the search space becomes linear to the plan length by interleaving planning and execution.
- The plan quality varies widely in RTA*, since the algorithm occasionally leads to a blind search.

[†8]Operator selection by DEE is a modification of that in STRIPS: $w\sum_{y_i} 1 - (1-w)\sum_{z_i} 1$. If w is near to 1, this formula means the following: first, select the operator that asserts the most subgoals; second, select the operator that deletes the least subgoals. The DEE formula gives considerable weight to the attainment of a subgoal with a high abstraction level value.

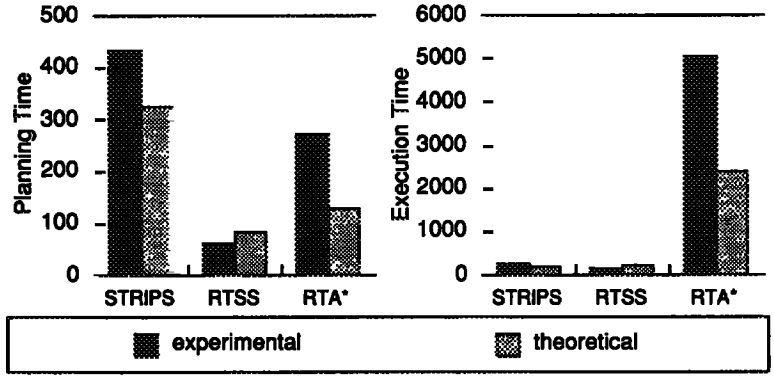

Figure 4: Comparison of Experimental and Theoretical Values of Planning Time and Execution Time ($d = 47.9$, $p = 0.510$ (RTA*), $p' = 0.879$ (RTSS), $q = 0.618$, $m = 10.7$, $u_R = 0.0545$, $u_S = 1.61$)

- The STRIPS's slope is steep, i.e. the cost for calculating one length plan is large. This is because of the domain-independent way of calculating operator instances, which requires the calculation of many instances.
- The execution time is small in STRIPS and RTSS, since DEE reduces the selection of superfluous operators.
- The plan's quality is good and the plan's variance is small in RTSS, since RTSS uses only an effective RTA* or STRIPS.

Figure 4 compares the theoretical and experimental performance values provided by the three algorithms. These values represent the average values for the solved problems. Each parameter was calculated from the experimental results. The probabilities p, p' and q were calculated based on the more accurate heuristic value of the heuristic function tuned for the robot's moving space. The times of subgoaling m was calculated by examining the plan length in a high abstraction level.

These profiles roughly resemble each other. This comparison indicates the following.

- The RTA* execution time critically changes when the probability p becomes close to 0.5. In the experiment, p often becomes close to 0.5. Therefore, the RTA* execution time is worse than those of the other algorithms.
- The probability q is 0.618. That this value is not bad is indicated by the STRIPS execution time. As mentioned above, this is because of operator selection by DEE.
- STRIPS's planning cost is considerably larger than that of RTA*. Their ratio is around 30 ($u_S/u_R = 30$).

Conclusions

This paper proposed an algorithm called Real-Time Search with Subgoaling (RTSS) which incorporates the STRIPS subgoaling function into RTA*. This algorithm overcomes the drawbacks of RTA* and STRIPS in real-time problem solving, i.e. the algorithm does not lead to a blind search in contrast to the other two. RTSS utilizes two indexes: degree of goal complexity (DGC) and degree of execution ease (DEE). The analysis of a simple model showed that the planning cost can be remarkably reduced by setting an appropriate threshold value for switching from subgoaling to real-time search. Furthermore, the experimental results of a robot task planning problem showed the effectiveness of RTSS. Our future work will examine alternative definitions of DGC and DEE and find a way of tuning the threshold for best performance.

Acknowledgments

The authors wish to thank Nick Short and Leonard Dickens for providing IDM (STRIPS-like planner); Seishi Nishikawa, Tsukasa Kawaoka, Ryohei Nakano and Nobuyasu Osato for their support in this work at NTT Laboratories; and Kazuhiro Kuwabara, Junichi Akahani and Makoto Yokoo for their helpful comments.

References

Drummond, M., Swanson, K., Bresina, J., and Levinson, R., "Reaction-First Search," *IJCAI-93*, pp.1408-1414, 1993.

Fikes, R.E, and Nillson, N.J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 2, pp.189-208, 1971.

Korf, R.E., "Real-Time Heuristic Search," *Artificial Intelligence*, 42, pp.189-211, 1990.

Sacerdoti, E.D., "Planning in a Hierarchy of Abstraction Space," *IJCAI-73*, pp.412-422, 1973.