A New Approach for Continual Planning (Extended Abstract)

Damien Pellier
Laboratoire d'Informatique de Paris Descartes
45, rue des St Pères
75006 Paris France
damien.pellier@parisdescartes.fr

Humbert Fiorino
Laboratoire d'Informatique de Grenoble
110 avenue de la Chimie
38400 Saint Martin d'Hères France
humbert.fiorino@imag.fr

Marc Métivier
Laboratoire d'Informatique de Paris Descartes
45, rue des St Pères
75006 Paris France
marc.metivier@parisdescartes.fr

ABSTRACT

Devising intelligent robots or agents that interact with humans is a major challenge for artificial intelligence. In such contexts, agents must constantly adapt their decisions according to human activities and modify their goal. In this extended abstract, we present a novel continual planning approach, called Moving Goal Planning (MGP) to adapt plans to goal evolutions. This approach draws inspiration from Moving Target Search (MTS) algorithms. In order to limit the number of search iterations and to improve its efficiency, MGP delays as much as possible the start of new searches when the goal changes over time. To this purpose, MGP uses two strategies: Open Check (OC) that checks if the new goal is still in the current search tree and Plan Follow (PF) that estimates whether executing actions of the current plan brings MGP closer to the new goal.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

General Terms

Algorithms

Keywords

Continual Planning, Moving Target Search

1. INTRODUCTION

In this extended abstract, we present a continual planning algorithm called MGP (Moving Goal Planning) built on the Moving Target Search (MTS) approach. MTS algorithms are search algorithms for real-time moving targets (an agent, "the hunter", follows a moving target, "the prey") interleaving pathfinding toward the prey and hunter displacements. MTS algorithms are based on heuristic search (distance calculation) and, to our knowledge, have not been used for task planning. It has long been considered difficult to find heuristic functions both informative and easily calculable for task planning. As a result, these two research areas have almost evolved independently. Thus, we propose to capitalize on recent advances in these two areas to devise new and efficient continual planners.

In highly dynamic environments, agents must constantly adapt the execution of their current plan to unforeseen events such as changes of goals. Many authors have considered this continual planning in two different ways [2, 11]: rebuilding a plan from scratch or repairing it so that it can be executed in the new context. Although in theory both approaches are equally expensive in the worst case [5], experimental results show that plan repair can produce repaired plans more efficiently than replanning from scratch [11]. Preserving plan stability is another argument in favor of the plan repair strategy [2].

In essence, a MTS algorithm interleaves pathfinding and action execution for a "hunter" agent chasing a moving target – the "prey" – over a large map or grid. The main strategy of MTS algorithms consists in reusing incrementally the search tree between two successive searches. The first algorithms based on this strategy are D* [6] and its successors [3, 9]. These algorithms were devised for replanning in unknown or changing environments and are both based on backward chaining. They perform correctly when the environment does not change much during the search. Otherwise, their performances are bypassed by simple successive calls to A* every time the target moves. As for FRA* [7], changes in the environment are not taken into account but it performs properly when the target moves over time. FRA* is based on the A* forward search. Every time the target moves, FRA* adapts quickly the search tree and recalls A* on the new search tree. FRA* is currently the most efficient MTS algorithm. However, the adaption of the search tree is widely dependent on the grid representation of the environment. In order to apply FRA* on more generic environments, a variant called GFRA* [8] has been recently proposed. Contrary to FRA*, GFRA* uses arbitrary graphs, including the state lattices used for Unmanned Ground Vehicles navigation. Another important feature of GFRA* is that it can be used with non admissible heuristics. Finally, Sun [10] proposed an algorithm called I-ARA*, which is the first incremental anytime search algorithm for moving target search. I-ARA* operates in the same way as repeated ARA* [4], except that it also uses incremental search as used in G-FRA* to speed up the search by adapting the tree search and by reusing the information from the previous search.

2. MOVING GOAL PLANNING

MGP addresses sequential planning in the propositional STRIPS framework [1]. All sets are finite. A *state* s is a set of logical propositions. An *action* a is defined as a tuple $a = (pre(a), add(a), del(a))$ where $pre(a)$ are the action *precondi-*

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

tions, $add(a)$ and $del(a)$ are respectively its positive and negative *effects*. The state s' is reached from s by applying an action a according to the transition function γ:

$$s' = \gamma(s,a) = \begin{cases} (s - del(a)) \cup add(a) & \text{if } pre(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases} \quad (1)$$

By extension, the application of a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ to a state s is recursively defined as

$$\gamma(s, \langle a_1, \ldots, a_n \rangle) = \gamma(\gamma(s, \langle a_1, \ldots, a_{n-1} \rangle), \langle a_n \rangle) \quad (2)$$

A Moving-Goal Pursuit problem is a tuple (A, s_t, g_t): at a given timestamp t, an agent is in a state s_t; g_t is its current goal (s_t and g_t are sets of propositions) and A is the set of actions that it can perform. It executes actions in order to reach its goal and the goal can change at any time. The agent has no information on how the goal changes over time. However, we assume that, at any time, the goal g_t is reachable: a *plan* is a sequence of actions $\pi_t = \langle a_1, \ldots, a_n \rangle$ ($a_i \in A$) such that $g_t \subseteq \gamma(s_t, \pi)$ and g_t is *reachable* if such a plan exists. A goal state is a state s such that $g_t \subseteq s$. At a given time t, a Moving-Goal Pursuit problem is solved if $g_t \subseteq s_t$ (the agent has reached its goal).

The MGP pseudocode is given in Algo. 1. MGP takes as input a Moving Goal Pursuit problem (A, s_0, g_0). The variables g and s denote respectively the current goal and the current state set initially to g_0 and s_0 (i is the search iteration counter).

MGP iterates a search procedure (line 2) as long as the current goal has not been reached. This procedure builds a search tree whose nodes are states and edges are actions. The search procedure fails if the current goal has not been reached and MGP fails (line 3). This is the case where the planning problem is unsolvable. Otherwise MGP postpones as much as possible a new search, i.e., a new expansion of the search tree (while-loop line 4) and it extracts a plan from the search tree built during the search (lines 5). The while-loop continues according to two delaying strategies:

- *Open Check (OC):* it checks if the new goal is still in the search tree. In that case, a new plan can be extracted in the current search tree.

- *Plan Follow (PF):* it estimates whether executing the actions of the current plan brings it closer to the new goal and determines if the current plan can still be used.

Thus, as long as the goal is reachable with the extracted plan or does not significantly change, MGP executes the actions of this plan (lines 8-9) and update it current state (line 10). The goal changes are simulated by the call to the procedure UpdateGoal line 11. Then, if MGP reaches its current goal, it returns success (line 12). Otherwise, MGP reduces its search tree to the subtree whose root is the current state s (DeleteStatesOutOfTree, line 13). If the new goal is in this subtree and a new search can be postponed, MGP extracts a new plan and executes its actions to reach the new goal. Otherwise, MGP updates the heuristic values of the search tree nodes according to the new goal (line 14) and expands this search tree (line 3).

Algorithm 1: MGP(A, s_0, g_0)

```
1  $s \leftarrow s_0, g \leftarrow g_0, i \leftarrow 1$
2  while $g \not\subseteq s$ do
3      if Search$(A, s, g, i)$ fails then return Failure
4      while OpenCheck$(g)$ and PlanFollow$(s, g)$ do
5          Extract a solution plan $\pi$ from the search tree
6          while ($g \not\subseteq s$ and $g \subseteq \gamma(s, \pi)$)
7                  or (PlanFollow$(s, g)$ and $\pi \neq \emptyset$) do
8              $a \leftarrow$ get and remove the first action of $\pi$
9              execute $a$
10             $s \leftarrow \gamma(s, a)$
11             $g \leftarrow$ UpdateGoal$(g)$
12         if $g \subseteq s$ then return Success
13         DeleteStatesOutOfTree$(s)$
14     UpdateSearchTree$(s, g)$
15     $i \leftarrow i + 1$
```

delays as much as possible starting new searches when the goal changes. To this purpose, MGP uses two search delaying strategies: Open Check (OC) that checks if the new goal is still in the search tree and Plan Follow (PF) that estimates whether executing the actions of the current plan brings MGP closer to the new goal.

3. CONCUSION

In this extented abstract, we have proposed a novel approach to continual planning, called MGP, which considers plan adaptation to constantly changing goals as a process pursuing "moving" goals. MGP is based on an incremental search and interleaves planning and execution when the goal changes over time. In order to limit the number of search iterations and to improve its efficiency, MGP delays as much as possible starting new searches when the goal changes. To this purpose, MGP uses two search delaying strategies: Open Check (OC) that checks if the new goal is still in the search tree and Plan Follow (PF) that estimates whether executing the actions of the current plan brings MGP closer to the new goal.

The MGP approach opens the way to several avenues of research. At first, it would be interesting to generate goal changes according to MGP heuristic function and domain-dependent strategies. In addition, a natural extension of this work would be to constrain MGP search and execution to time limits and to address real-time continual planning.

4. REFERENCES

[1] R. Finke and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3-4(2):189–208, 1971.

[2] M. Fox, A. Gerevini, D. Long, and I. Serina. Plan stability: Replanning versus plan repair. In *Proc. ICAPS*, pages 212–221, 2006.

[3] S. Koenig and M. Likhachev. D* lite. In *Proc. AAAI*, pages 476–483, 2002.

[4] M. Likhachev, M. Gordon, and S. Thrun. ARA*: Anytime a* with provable bounds on sub-optimality. In *Proc. NIPS*, 2003.

[5] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454, 1995.

[6] A. Stenz. The focused D* algorithm for real-time replanning. In *Proc. IJCAI*, pages 1642–1659, 1995.

[7] X. Sun, W. Yeoh, and S. Koenig. Efficient incremental search for moving target search. In *Proc. IJCAI*, pages 615–620, 2009.

[8] X. Sun, W. Yeoh, and S. Koenig. Generalized Fringe-Retrieving A*: Faster Moving Target Search on State Lattices. In *Proc. AAMAS*, pages 1081–1088, 2010.

[9] X. Sun, W. Yeoh, and S. Koenig. Moving target D* lite. In *Proc. AAMAS*, pages 67–74, 2010.

[10] X. Sun, W. Yeoh, T. Uras, and S. Koenig. Incremental ARA*: An Incremental Anytime Search Algorithm for Moving Target Search. In *Proc. ICAPS*, 2012.

[11] R. van der Krogt and M. de Weerdt. Plan repair as an extension of planning. In *Proc. ICAPS*, pages 161–170, 2005.