Moving Target Search with Subgoal Graphs*

Doron Nussbaum and **Alper Yörükçü**
School of Computer Science
Carleton University

Abstract

Moving Target Search (MTS) is a dynamic path planning problem, where an agent is trying to reach a moving entity with a minimum path cost. Problems of this nature can be found in video games and dynamic robotics, which require fast processing time (real time). In this work, we introduce a new algorithm for this problem - the Moving Target Search with Subgoal Graphs (MTSub). MTSub is based on environment abstraction and uses Subgoal Graphs to speed up the searches for a minimal cost route to the target. The algorithm is optimal with respect to the knowledge that the agent has during the search. Experimental results show that MTSub can be used in real-time applications (e.g., applications requiring 5 microseconds response time per step). The experiments compared MTSub to G-FRA*, which is the best known dynamic algorithm so far, showing that MTSub is up to 29 times faster in average time per step, and up to 186 times faster in maximum time per step.

Introduction

A large number of applications, in video games, robotics and virtual simulations, require agents to plan their path, not only to a stationary target but also with respect to a moving target. Namely, when the target moves to a new location, agents must modify their route during execution and compute a new route to the new target location. The speed with which these operations are executed is important because agents must determine the next move in real time. The quality of the updated route is also crucial because it directly affects the cost of reaching the target.

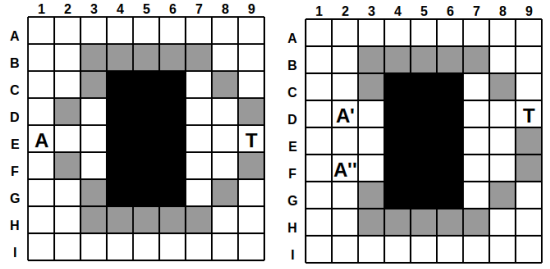
Moving Target Search (MTS) is a path planning problem where an agent attempts to reach a moving target (Ishida and Korf 1991). In this paper, we focus on moving target search where at all times, the agent has full knowledge of the search environment, which is static, and the target position. This variant of MTS often arises in robotic applications that run in a known environment and in video games.

*The research is partially supported by the Natural Sciences and Engineering Research Council of Canada.

There are generally two kinds of strategies for solving MTS problems with search algorithms (Sun, Yeoh, and Koenig 2010):

- *Offline techniques* (Vieira, Govindan, and Sukhatme 2008; 2009) consider all possible locations of the agent and the target in the environment to determine the best plan, prior to movement of the agent.
- *Online techniques* (Ishida and Korf 1991; Sun, Yeoh, and Koenig 2009; 2010) find a solution according to current information and update the existing plan when changes occur. They solve a series of path planning problems during plan execution to react to the target movement.

We introduce an innovative, incremental search algorithm that uses environment abstraction to shorten response time of the agent without giving up cost minimal paths, Moving Target Search with Subgoal Graphs (MTSub) (Nussbaum and Yörükçü 2015). MTSub uses Subgoal Graphs (Uras, Koenig, and Hernández 2013) to create an abstract search environment and determine the path by solving numerous instances of the path planning problem incrementally. Subgoal Graphs restricts the applicability of MTSub to eight connected grids but this also lets MTSub to exploit environment specific features.



(a) Positions at time t_i (b) Positions at time t_{i+1}

Figure 1: a. Two possible paths, with the same cost, that the agent, A, can take to reach the target T at time t_i; b. Positions of A and T at time t_{i+1} after A chose one of the two paths randomly.

The problem is defined as follows: Given a graph $G = (V, E)$, representing an 8-connected environment, an agent A and a moving target T, positioned at s_A and s_T respectively, find an optimal route that is subject to:

- A and T can move at discrete time steps.

- At each time step t_i, T positioned at $u = position(T, t_i)$ can either stay put or move to node v where $v \in neighbour(u)$.
- At each time step t_i, A, positioned at ($p = position(A, t_i)$), can detect the position of T ($u = position(T, t_i)$), compute a path $\Pi(p, u)$ and move to a node w where $w \in neighbour(u) \wedge w \in \Pi(p, u)$.

An optimal route is a route from $position(A, t_o)$ to $position(T, t_k)$, when t_o is the time that A and T attempted to move, t_k is the time that A reached T, and $\forall t_i\ t_o \leq t_i \leq t_k$, $position(A, ti) \in\ \Pi(position(A, t_{i-1}), position(T, t_{i-1}))$ where Π is a cost minimal path.

Note that following an optimal route may not minimize the cost of reaching the target. Figure 1 depicts a search environment at time t_i and t_{i+1} where black cells represent obstacles, 'A' and 'T' show s_A and s_T, respectively. Two obstacle free paths are displayed in Figure 1a. The paths have the same cost. However, Figure 1b shows that the agent's random choice of paths at time t_i may not be the best choice to get closer to the target. Experiments show that following an optimal route decreases the cost of reaching the target in most cases (Sun et al. 2012).

Moving Target Search with Subgoal Graphs

Moving Target Search with Subgoal Graphs(MTSub) is an optimal algorithm that attempts to update the path based on knowledge from the previous step. MTSub is different from the other incremental MTS algorithms because it uses a Subgoal Graph (SG) (Uras, Koenig, and Hernández 2013), which is constructed during a preprocessing step.

MTSub used SG to create environment abstraction, thus, reducing the size of the search space of the eight connected grid environment. Instead of searching, at every time step, for a cost minimal path in the given eight connected graph, MTSsub searches the SG. MTSub achieves its speed up by incremental search of the environment using the SG and by reducing the number of times that a complete shortestp path tree is computed.

We compared MTSub against several MTS algorithms: G-FRA*, which is an incremental algorithm, and A* and Two-Level Subgoal Graphs which are shortest path planning algorithms. The latter two algorithms are invoked repetitively to solve MTS and are referred to as Repeating A* (R-A*) and Repeating Two-Level Subgoal Graphs (R-Sub).

The experiments were conducted as follows: First, a predefined set of maps were selected. The size of a graph and obstacles size and location were determined by each map. Second, 100 scenarios were created including target and agent initial location, and the target's random motion (each algorithm was tested using the same 100 scenarios). Four classes of maps were used to examine the impact of different environment on the algorithms performance: constructed, maze, random and room maps. We conducted the experiments in maps scaled 512 x 512 with 100 scenarios per map. For each map class, we chose 6 different maps.

The results of the experiments are depicted in Table 1. In these experiments MTSub outperforms all of the competing algorithms in both average and maximum time per step. In comparison to G-FRA*, MTSub is up to 29 times faster in average time per step, and up to 186 times faster in maximum time per step. The results shows MTSub advantage of using environment abstraction and incremental search.

		R-A*	R-Sub	GFRA*	MTSub
Constructed	Avg.	325.20	12.09	140.54	6.85
	Max	9369	188	26857	144
Random	Avg	284.2	270.10	91.66	50.56
	Max	5389	6588	15533	5140
Room	Avg.	626.48	23.09	163.25	5.66
	Max	9858	810	33507	304
Maze	Avg.	8571.96	852.71	648.5	44.85
	Max	38315	7871	80054	4523

Table 1: Experiments in 512 x 512 Maps, Times are Given in μ seconds. All reported times are per step where Avg. is the average time per step and Max is the maximum time per step. Note the effect of map type on the performance of the algorithms.

Conclusions

In this paper, we introduced an innovative incremental search algorithm that uses environment abstraction to speed up the response time of an agent that searches for a moving target. Empirical evaluation shows that MTSub outperforms G-FRA*, R-Sub and R-A* in all of the experiment domains. MTSub average performance meets real time requirements and is applicable to video games and robotics that use not only small environments but also larger ones with lower resources.

References

Ishida, T., and Korf, R. E. 1991. Moving target search. In *IJCAI*, volume 91, 204–210.

Nussbaum, D., and Yörükçü, A. 2015. Moving target search with subgoal graphs. In *Twenty-Fifth International Conference on Automated Planning and Scheduling. Forthcoming.*

Sun, X.; Yeoh, W.; Uras, T.; and Koenig, S. 2012. Incremental ara*: An incremental anytime search algorithm for moving-target search. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 243–232.

Sun, X.; Yeoh, W.; and Koenig, S. 2009. Efficient incremental search for moving target search. In *Twenty-First International Joint Conference on Artificial Intelligence*, 615–620.

Sun, X.; Yeoh, W.; and Koenig, S. 2010. Moving target d* lite. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 67–74. International Foundation for Autonomous Agents and Multiagent Systems.

Uras, T.; Koenig, S.; and Hernández, C. 2013. Subgoal graphs for optimal pathfinding in eight-neighbor grids. In *Twenty-Third International Conference on Automated Planning and Scheduling*, 224–232.

Vieira, M. A.; Govindan, R.; and Sukhatme, G. S. 2008. Optimal policy in discrete pursuit-evasion games. *Department of Computer Science, University of Southern California, Tech. Rep* 08–900.

Vieira, M. A.; Govindan, R.; and Sukhatme, G. S. 2009. Scalable and practical pursuit-evasion with networked robots. *Intelligent Service Robotics* 2(4):247–263.