Optimal Solutions for Moving Target Search

(Extended Abstract)

Carsten Moldenhauer
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
moldenha@cs.ualberta.ca

Nathan R. Sturtevant
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
nathanst@cs.ualberta.ca

ABSTRACT

Moving target search or the game of cops and robbers has been given much attention during the last two decades. It is known that optimal solutions, given a n-cop-win graph, are computable in polynomial time in the size of the input graph. However, a first practical polytime algorithm was only given recently by Hahn *et al.* [3]. All other known approaches are learning and anytime algorithms that try to approximate the optimal solution. In this work we present four algorithms: an adaptation of Two-Agent IDA*, Proof-Number Search, alpha-beta, and Reverse Minimax A*, a new algorithm. We show how these techniques can be applied to compute optimal moving target search solutions and give benchmarks on their performance for the one cop one robber problem.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms

Keywords

Moving target search, cops and robber

1. INTRODUCTION

The game of cops and robbers was first introduced by Quillot [10] and Nowakowski *et al.* [8]. It was introduced into the AI literature by Ishida *et al.* [5] as a new variation of the general search problem, moving target search (MTS). As the name suggests, the goal of this game is for a number of cops, controlled by one player, to cooperate and catch a robber, controlled by the second player. All agents are assigned an initial position in a given map before the start of the game. Both players move alternately with the cop player beginning. A move consists of choosing new locations for each of the agents under a player's control. An agent can either move to any neighbor of its current location or *pass* and remain in the same location. No two cops are allowed to occupy the same vertex at any time. The goal of the cop player is to eventually move one of his cops onto the same vertex as the robber, thus capturing it. The robber's goal is to evade capture as long as possible. The game is played with perfect information, *i.e.* both players know all agent positions at all times.

A graph is said to be n-cop-win if n cops have a winning strategy, *i.e.* a strategy that eventually captures the robber. Otherwise, the robber can evade capture forever. Given a fixed number of cops n, it is possible to decide in polynomial time (on the size of the input graph) if the robber can be caught. A first polytime algorithm was given by Hahn and MacGillivray [3]. This immediately gives rise to the question of how to compute optimal solutions quickly.

Despite recent research on MTS, there has been no systematic study of optimal algorithms. There are only a variety of approaches of learning and any-time algorithms that yield good solutions but none are guaranteed to return optimal solutions [6, 2, 7, 4].

Cite as: Optimal Solutions for Moving Target Search, (Extended Abstract), Carsten Moldenhauer, Nathan R. Sturtevant, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1249–1250

2. APPROACHES

We introduce three methods for computing optimal solutions for moving target search. We use admissible heuristics to speed up computation. Given an initial state s, let $h^*(s)$ denote the value of a game (moves until capture) obtained by optimal play starting in s. Then, an algorithm is admissible if it computes $h^*(s)$ and a heuristic h is admissible if $h(s) \leq h^*(s)$ for all states s. The following algorithms are admissible for any type of graph with any number of cops when used with admissible heuristics. However, we only experiment on graphs that are 1-cop-win. There are specialized algorithms that are optimal for certain types of graphs and for certain numbers of cops, but they are not considered here.

2.1 Two-Agent IDA*

Two-Agent IDA* (TIDA*) was developed by Prieditis and Fletcher [9], however it is not widely-known. The algorithm is designed for two player adversarial games where players want to minimize or maximize accumulated transition and terminal costs. The basic idea is to maintain a lower bound on the value of the root of the game tree. In each iteration this bound is increased until no improvement can be made. Heuristic bounds are used to prune the search space in a similar way to how the single-agent IDA* algorithm prunes nodes with high costs.

The cops and robber domain has these properties, in that the goal is to maximize/minimize the distance to capture, which can be estimated with a heuristic. TIDA* normally uses an artificial depth-limit to prune the search when time is limited and only an approximate solution is needed. We remove this limit and add caching to the algorithm to handle transpositions in the game tree. Without this enhancement, it would not be a feasible approach.

2.2 Proof-number Search

Proof-number search was developed by Allis *et al.* [1] to prove or disprove a lower bound on the root's value. Instead of perform-

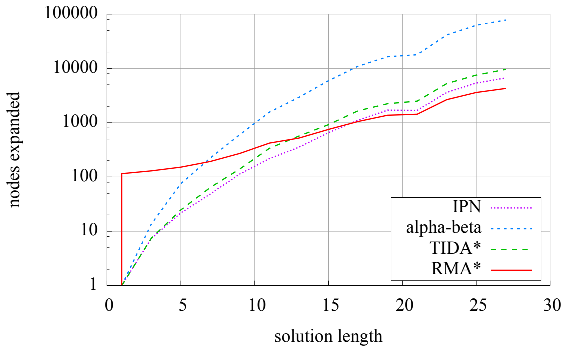

Figure 1: Performance of IPN, α-β, TIDA* and RMA* on 15x15 maps.

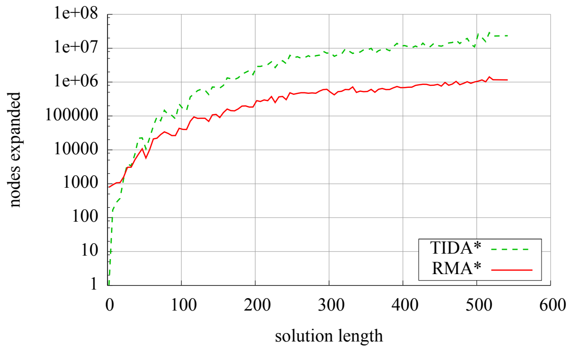

Figure 2: Performance of TIDA* and RMA* on 40x40 maps.

ing binary search on such bounds to find the correct value, we use the iterative bound increase known from TIDA*, hence the name *iterative proof-number search* (IPN).

The main drawback of IPN is space complexity because the computed game tree has to be kept in memory. We implemented a version of IPN that keeps only one node per state per depth in the tree. Therefore, polynomial space complexity is gained. However, updating the (dis)proof numbers of the parents is more complicate and thus more expensive.

2.3 Reverse Minimax A*

Two-player game-tree search usually begins from a starting position and works towards possible goal positions. Since the cops and robber problem has many possible goal positions, we develop Reverse Minimax A* (RMA*) which generates the game tree from the leaf nodes towards the root. This is a similar approach to building an endgame database in Checkers [11]. A first dijkstra-like algorithm has been developed by Hahn and MacGillivray [3] to generate optimal values for the entire state space. In contrast, we use this algorithm to search for one particular given initial state and incorporate a heuristic function to speed up the search.

3. EXPERIMENTAL RESULTS

Within our experiments, we tested on the one cop one robber problem on maps with octile connections that induce 1-cop-win graphs. We set all edge costs and the cost for passing a turn to one. As admissible heuristic, we used a maximum norm distance metric.

We tested maps of size 15x15 (set one) and 40x40 (set two). Set one contains 15 maps and set two contains 20 maps. On each map, we generated 50 random initial locations for the cop and the robber.

Besides the above algorithms, we also compare performance of the α-β algorithm enhanced with heuristic pruning and transposition tables. We measure the number of node expansions with respect to solution length, *i.e.* the depth of the game tree that has to be computed. Plots with logarithmic scale on the number of node expansions can be found in Figures 1 and 2.

The α-β does not scale well to large map sizes (Figure 1). Although transposition tables are used, the number of node expansions seems to remain exponential in the depth of the computed game tree. IPN expands fewer nodes than TIDA*. This is expected since IPN is designed to expand as few nodes as possible to quickly (dis)prove the root's value. However, due to required updates of the (dis)proof numbers, IPN touches an order of magnitude more nodes than any other algorithm (plot omitted due to space restrictions), thus renders itself impractical.

Large maps can only be solved by TIDA* and RMA*. It can be seen from Figure 2 that TIDA* expands an order of magnitude more nodes than RMA*. However, due to the overhead of keeping a sorted open list and having to push all the terminal states on the list before starting the search, this does not pay off for small solution lengths. Therefore, when dealing with small solution lengths in large maps, it is more promising to use TIDA*. For long solutions lengths RMA* is the method of choice.

4. REFERENCES

[1] L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artificial Intelligence*, 66:91–124, 1994.

[2] M. Goldenberg, A. Kovarksy, X. Wu, and J. Schaeffer. Multiple agents moving target search. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[3] G. Hahn and G. MacGillivray. A note on k-cop, l-robber games on graphs. *Discrete Mathematics*, 306(19-20):2492–2497, 2006.

[4] A. Isaza, J. Lu, V. Bulitko, and R. Greiner. A cover-based approach to multi-agent moving target pursuit. *Artificial Intelligence and Interactive Entertainment Conference (AIIDE)*, 2008.

[5] T. Ishida and R. Korf. Moving target search. *International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 204–210, 1991.

[6] T. Ishida and R. E. Korf. Moving-target search: A real-time search for changing goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):609–619, 1995.

[7] S. Koenig, M. Likhachev, and X. Sun. Speeding up moving-target search. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007.

[8] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43:235–239, 1983.

[9] A. E. Prieditis and E. Fletcher. Two-agent ida*. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(4):451–485, 1998.

[10] A. Quillot. *Etudes de quelques problèmes sur les graphes et hypergraphes et applications à la théorie des jeux à information complète*. PhD thesis, Université de Paris VI, 1978.

[11] J. Schaeffer, N. Burch, Y. B. Zrnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, pages 1518–1522, 2007.