Multiple Agents Moving Target Search

Mark Goldenberg, Alexander Kovarsky, Xiaomeng Wu, Jonathan Schaeffer
Department of Computing Science, University of Alberta,
Edmonton, Alberta, Canada T6G 2E8
{goldenbe,kovarsky,xiaomeng,jonathan}@cs.ualberta.ca

Abstract

Traditional single-agent search algorithms usually make simplifying assumptions (single search agent, stationary target, complete knowledge of the state, and sufficient time). There are algorithms for relaxing one or two of these constraints; in this paper we want to relax all four. The application domain is to have multiple search agents cooperate to pursue and capture a moving target. Agents are allowed to communicate with each other. For solving Multiple Agents Moving Target (MAMT) applications, we present a framework for specifying a family of suitable search algorithms. This paper investigates several effective approaches for solving problem instances in this domain.

1 Introduction

In the 2002 Steven Spielberg movie *Minority Report*, John Anderton (played by Tom Cruise) is on the run. In one sequence, Anderton is hiding in a building, and his pursuers unleash a team of mini-robots to flush him out. The robot team separates, each covering a different part of the building. Anderton, realizing the danger, stops fleeing and comes up with a unique solution – he submerges himself in a bathtub of water so as to avoid the robotic detectors. Sadly, he can only hold his breath for so long, before he has to emerge and is found by the robots.

The classic algorithms (such as A^*) are effective for solving search problems that satisfy the properties: one search agent, the agent has perfect information about the environment, the environment and goal state do not change, and enough time is given to make an optimal decision. Relaxing even one of the assumptions gives rise to new algorithms (e.g., moving target search [Ishida and Korf, 1995]), real-time search [Korf, 1990], and D^* [Stentz, 1995]). Many real-world problems do not satisfy all of these properties. In this paper we use an application domain that breaks all of them.

Consider the task of multiple agents having to pursue and capture a moving target; for example a squad of policemen chasing a villain. We want to design a test environment that is as realistic as possible. We assume a grid with obstacles. Agents can only "see" what is directly visible to them. Agents are allowed to communicate with any agent that they can see. As the target flees, it may become obscured from sight.

The agent's knowledge of the locations of the target and other agents may be fuzzy (since some of them may be hidden from sight). Given whatever knowledge of the target and other agents is available, an agent must decide how to pursue the target. The target's possible locations provide information of where to search; the other agents' possible locations provide information that can be used to coordinate the search effort. The challenge is to have the agents act autonomously to catch the target as quickly as possible.

This paper makes the following contributions: MAMT – a challenging application domain for exploring issues related to Multiple Agents pursuing a Moving Target, a framework for expressing real-time search algorithms for the MAMT domain, and several solutions that allow an agent to act autonomously (using information about the possible positions of the target and other agents in the decision-making process).

More details are available at `www.cs.ualberta.ca/~jonathan/Papers/ai.2003.html`.

2 Literature

There are a family of A^* algorithms that relax solution optimality by requiring the agent to make the best decision possible given limited search resources (e.g., time) [Korf, 1990]. The *Minimin Lookahead Search* algorithm uses a fixed-depth search (d moves), keeping track of the moves that lead to the (heuristically) best d-move outcome. *Real-Time* A^* (RTA^*) is an A^* variant that uses the results produced by the minimin lookahead search as heuristic values in order to guide the search towards achieving the goal [Korf, 1990]. *Moving Target Search* (MTS) is an $LRTA^*$ variant that allows a moving target [Ishida and Korf, 1995]. An assumption in most *MTS* papers is that the target moves slower than the agent. Without this requirement, the target can stay ahead of the agent and possibly elude capture. There are many real-time search variants (e.g., LPA* [Koenig and Likhachev, 2003]). For the most part, these are orthogonal to our work. The difficult part for an agent is deciding on the search goal; once achieved, then any of several different search algorithms can be used.

Having multiple agents participating in a search is a topic of recent interest. RoboCup is an example, but that work has more limited scope, and agents (players) generally have global knowledge.

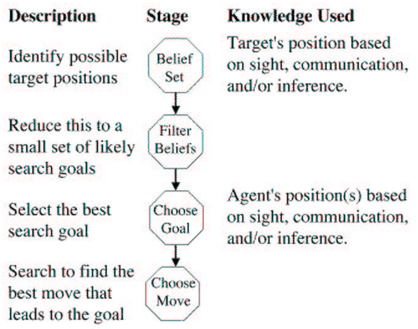
Description
Stage
Knowledge Used
Identify possible target positions
Belief Set
Target's position based on sight, communication, and/or inference.
Reduce this to a small set of likely search goals
Filter Beliefs
Select the best search goal
Choose Goal
Agent's position(s) based on sight, communication, and/or inference.
Search to find the best move that leads to the goal
Choose Move

Figure 1: Framework for MAMT solutions

3 Problem Description

The MAMT domain has the following properties. **Agents and target**: multiple agents pursuing a single moving target. **Grid**: $m \times n$ in size, with randomly-placed obstacles. All agents and the target have complete knowledge of the grid topology. **Moving**: all moves are horizontal or vertical and are made simultaneously. **Starting position**: The target always starts in the middle of the grid. The agents are all placed in the lower left corner of the grid. The target is visible to at least one agent. **Vision**: can "see" anything that is in an unobstructed direct line. **Communication**: between moves, agents communicate with any agent that is visible to them. The agents exchange information as to where they believe the target and other agents are located. **Objective**: catch the target in the fewest number of moves.

4 Multiple Agent Moving Target Search

The intent of this work is not to build a new search algorithm. Rather, we want to plug standard search algorithms into a framework that, given a goal selection, will find the "best" way to reach the objective.

When an agent does not know the exact position of the target (from vision or communication), it must maintain a *belief set* of where the target might be. The belief set can take into account the topology of the grid, knowledge of the opponent, and the time since the last known target location. As the time increases since the last sighting, the knowledge of where the target is gets fuzzier. An agent should choose its search area based on its beliefs about the target and other agents.

Figure 1 shows the four-step method that is the framework used for specifying our solution algorithms. There is no "right" way of solving any of these steps. In the following we detail several algorithm alternatives.

4.1 Belief Set

Whenever an agent knows the exact location of the target, that agent's belief set contains only one location, otherwise it can grow. Since this is a real-time search application, and real agents have limited memory, the belief set size is limited. Before a move, each agent sends its belief set information about the target and other agents to any agent it can see (who may, in turn forward it to agents that they see).

We implemented three strategies for maintaining the belief set. **All-scenarios.** Expand the current belief set to include all possible locations that can be reached in one more move. **Region belief set.** This set has the same update as the all-scenarios belief set. After the search goal is chosen, the agent *commits* to only consider beliefs that are connected to the goal location. **Single-location:** The agent maintains a single belief (one grid square). The belief is updated by choosing a random direction and moving the belief in that direction until an obstacle is encountered or new target information is available.

We use a simple greedy algorithm to approximate the four "corners" of the variable shaped belief set. This subset of the belief set is called the *filtered belief set*.

4.2 Goal Selection

Each agent selects a goal from their filtered belief set. Randomly selecting a location from this set is an obvious control strategy to implement. However, a more intelligent strategy is needed – one that considers information about other agents.

The *difference metric* is used to identify a goal that ideally is (a) closest to the agent and yet (b) farthest from the closest other agent. For each location in the filtered belief set, we compute two metrics: the distance from the agent to the belief, and the minimum distance from the belief to the last known agent positions. These values can be determined by search (expensive) or by heuristics (inaccurate).

For each location in the filtered belief set, the agent computes the difference of the above two values, and chooses the one with the minimum difference. The idea is that the agent should assist the other agents by covering the possible escapes of the target that are hard for the other agents to reach.

4.3 Search

Given a goal, each agent performs a search to find a "best" move that progresses towards that goal. Since this has to be a real-time algorithm, the search algorithm is allocated a fixed number of search nodes (approximating a fixed amount of time per decision). The manhattan distance is used as the search evaluation function.

We experimented with two search algorithms. **Single-agent minimin search** [Korf, 1990]: Here the agent uses single-agent search to find the shortest path to the goal location. The search has the effect of selecting the move that tries to chase the target. **Adversarial search**: The search can alternate between moves for the agent (move towards the target) and target (move away from the agent). This becomes an alpha-beta search, where the agent tries to minimize the minimax value of the search (the distance from the target).

4.4 Comments

For non-trivial belief sets, as long as information is known about other agent's positions the agents will separate to avoid redundancy in the search. Each agent wanders about the grid trying to maximize their coverage, as a function of what they know (about the target and other agents). In many cases (especially for large mazes) an agent may go a long time without getting an update on the target's position, effectively negating the effectiveness of the belief set.

5 Experiments

There are a variety of target strategies that can be investigated. The strategy used is a weighted combination of four

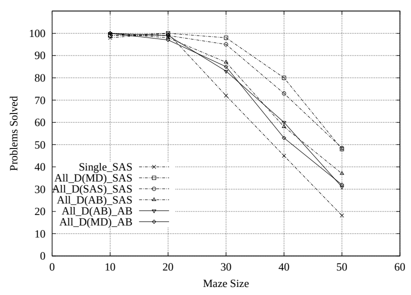

Figure 2: Comparing solutions

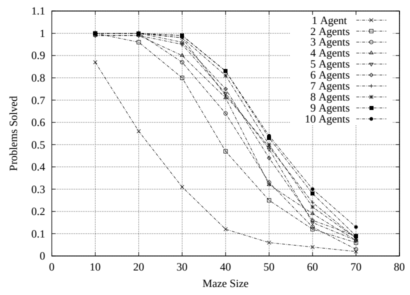

Figure 3: Varying size and # of agents

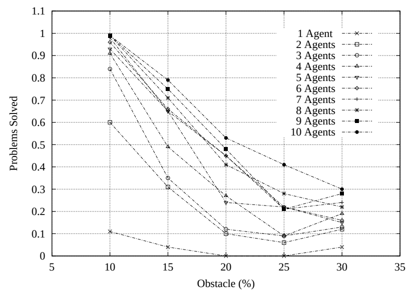

Figure 4: Varying the obstacle density

sub-strategies: distance, mobility, visibility, and random. The maximum scoring move is selected. The weights were hand-tuned based on the perceived realism of the target's behavior.

Solutions to MAMT instances were tested using grids of sizes from 10×10 to 70×70. Grids had obstacles randomly placed, occupying 10% to 30%, in increments of 5%, of the space. The experiments used 1 to 10 agents. Each pursuer was allocated 50,000 search nodes to make its decision. If search was used to compute the difference metric, then half the search nodes were allocated to this task, and the other half to move selection. Agents were allowed a maximum belief set size of 100. An experiment ended when one of the pursuers caught the target, or when a maximum number of moves was reached. For an $n \times n$ grid, the maximum was set to $15 \times n$. With few exceptions, the target was caught in less than $8 \times n$ moves, or not at all.

Figure 2 compares several different solutions. The belief set was maintained using one of: all-scenarios (ALL), region, and single-location (Single). Except for a single-location belief set, the choice of goal was done using the difference metric based on either manhattan distance (no search) – D(MD), single-agent search – D(SAS), or alpha-beta – D(AB). Having chosen a goal, single-agent search (SAS) or alpha-beta (AB) was used to select the best move. The graph shows the percentage of problems where the target was caught (100 trials with different random seeds per data point) as a function of the maze size.

The control experiment is randomly selecting a goal (single-location), and using single-agent search (SAS) to decide on the move choice. Not surprisingly, this gets poor performance. Region was expected to perform quite well, but instead its results were mixed (not shown).

All_D(MD)_SAS and All_D(SAS)_SAS performed best in all our experiments, with a preference for the manhattan difference metric. That the difference heuristic gets the best performance is gratifying, since it is better informed by using beliefs about the other agents. Using a simple heuristic appears to be as good as or better than using search for determining the "best" search goal. The simplest way of computing the difference metric – using static manhattan distance instead of more accurate search – also leads to shorter solution lengths (up to 10% on average). This shows that it is more beneficial to invest search effort in move selection than in goal selection.

Alpha-beta is out-performed by single-agent search. Since alpha-beta takes into account the target's moves, it cannot reach the search depths that single-agent search can and, hence, usually yields a lesser-quality solution.

In cases where the target eluded capture, a familiar pattern emerged. The target would stay hidden in a small area, and the agent's knowledge of where the target was became obsolete and effectively useless. The agents would independently wander about, hoping to find the target. In the real world, if such a scenario arises, the agents should wait until more help arrives and then begin the search anew, going through the entire grid systematically.

Figure 3 shows that more agents are better than fewer (using All_D(MD)_SAS). Note that as the number of agents increases, the number of nodes per agent per search in the goal selection gets smaller (recall the *total* search size is limited). Even so, adding more agents is beneficial despite less resources available.

Figure 4 shows that as the mazes become more congested with obstacles, it gets harder for the agents to find the target. Essentially, a higher percent of obstacles gives the target more opportunities to hide. However, at 30% of obstacles, our target starts having problems with avoiding the dead ends and is sometimes caught more easily.

Several control experiments were also done, and they gave predictable results. Simplistic targets (e.g., random, or avoid) were much easier to catch.

6 Future Work and Conclusions

This problem domain is rich in possibilities, and can be extended to increase the "realism" of the simulations. Some examples include: multiple moving targets, more realistic communication, creating "human-like" target behavior, and opponent modeling. The framework used of this research has many opportunities for interesting extensions.

7 Acknowledgments

This research was supported by NSERC, iCORE, and IRIS.

References

[Ishida and Korf, 1995] T. Ishida and R. Korf. Moving target search: A real-time search for changing goals. *IEEE PAMI*, 17(6):609–619, 1995.

[Koenig and Likhachev, 2003] S. Koenig and M. Likhachev. D* lite. *AAAI*, pages 476–483, 2003.

[Korf, 1990] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2–3):189–211, 1990.

[Stentz, 1995] A. Stentz. The focussed D* algorithm for real-time replanning. In *IJCAI*, pages 1652–1659, 1995.