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

April 11, 2003

Abstract

For path-finding problems, traditional single-agent search algorithms usually make simplifying assumptions. These include having a single search agent, a stationary target, complete knowledge of the state, and sufficient time to make an optimal decision. 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 (e.g., squad of policemen) cooperate to pursue and capture a moving target (e.g., the villain). Agents are allowed to communicate with each other, but only if they are visible to 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. Anderton is being chased by his own team of detectives, whose job it is to identify "pre-crimes" and arrest the guilty party before they commit the crime. In one sequence, Anderton is hiding in a building, and the pre-crime investigators 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 above scenario has several aspects that are not handled by the conventional artificial intelligence search algorithms. The classic algorithms, such as A^* and IDA^*, provide simple and effective approaches for solving search problems that satisfy the following properties:

1. One search agent.

2. Agent has perfect information about the environment.

3. Environment and the goal state do not change.

4. Enough time is given to make an optimal decision.

Relaxing even one of the assumptions gives rise to new algorithms. For example, allowing the target to move (moving target search [3]), limiting the time allowed to make a decision (real-time search [5]), and dynamically changing the search topology (e.g., D^* [9]) have spawned interesting research directions. Many real-world problems do not satisfy all of the above four properties and in this paper we deal with 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 – other agents or the target. 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 to obtain maximum coverage.

The challenge is to have the agents act autonomously to catch the target as quickly as possible. Coming up with effective algorithms for this problem domain has obvious applications to police or military simulations. However, the motivating application for this technology is commercial computer games. Many commercial games products are still grappling with the computational complexity of using A^* variants in real time [10]; the (non-cheating) coordinated actions of multiple agents pursuing a target represents next-generation technology. Recently there have been efforts to apply commercial game technology to creating virtual reality training programs (e.g., [1]). The technology in this paper could be used to create realistic police chase scenarios for training new recruits.

For this application domain, which we call Multiple Agent Moving Target (MAMT), we define a framework for specifying algorithmic solutions. There are a wide variety of possible solutions, several of which are presented in this paper. Designing an algorithm that achieves "realistic" pursuit behavior turns out to be a challenging problem.

This paper makes the following contributions:

1. MAMT – a challenging application domain for exploring issues in real-time search.

2. A framework for expressing a real-time search algorithm for MAMT domains.

3. Several solutions that allow an agent to act autonomously, using information about the possible positions of the target and other agents as part of the decision-making process.

Section 2 discusses the related literature. Section 3 defines the MAMT problem domain used in this paper. Section 4 introduces the framework and some algorithmic instances. Section 5 evaluates different solutions, and Section 6 presents ideas for future research.

2 Literature

There are a family of A^* algorithms that relax the need for an optimal answer by requiring the agent to make the best decision possible given limited search resources (e.g., time) [5]. 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. The heuristic values are propagated up the tree, with α-pruning used to reduce the search effort. The minimum f-value of the leaf nodes seen so far is called α. A cut-off occurs whenever the heuristic value of an interior node is $\geq \alpha$.

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 [5]. The algorithm maintains a hash table with the results of the previous lookahead searches. The agent always moves to the state that is the closest to the (fixed) goal according to a value in the hash table or (if the corresponding value is not in the hash table) according to the result of the lookahead search. After that, the second best value of the lookahead searches performed at the last state is written to the hash table. In *Learning Real-Time* A^* *(*$LRTA^*$*)*, the best f-value is written to the hash table instead of the second best. This change avoids inflated values in the hash table in the repeated trials and guarantees convergence of the algorithm to the perfect heuristic.

The *Moving Target Search* (MTS) algorithm is an $LRTA^*$ variant that allows a moving target [3]. Enhancements to this basic idea include adding notions of commitment and deliberation to the agent [3, 2]. 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, in many cases, elude capture.

There are many variants of real-time search, including recent additions addressing issues such as re-planning (e.g., Lifelong Planning A* [4]). However, these are for the most part orthogonal to our work. The difficult part for an agent in the MAMT domain is deciding on the search goal; after that is achieved, then any of several different search algorithms (such as outlined above) can be used.

Having multiple agents participating in a search is a topic of recent interest. For example, the RoboCup teams have multiple cooperating agents with communication. In [8], different agent world models are discussed and evaluated. However, their work assumes that more state information is available than would be in a MAMT-like problem domain.

In the area of commercial games, work has progressed on developing solutions for related applications. Research on *squad tactics* [11] and *tactical teams* [7]) concentrates on having cooperating multiple agents follow prescribed military strategies, and is not suitable for pursuit.

3 Problem Description

The following describes the experimental domain used for this research. The domain can be extended to be more realistic. However, even the "simplified" domain used here is quite challenging. Section 6 describes some extensions.

The MAMT domain has the following properties.

1. Agents and target. There are multiple agents pursuing a single moving target.
2. Grid. The grid is $m \times n$ in size, with obstacles randomly placed. All agents and the target have complete knowledge of the grid topology. We allow this assumption because that information is typically available in commercial computer games. This assumption can be relaxed (see Section 6).
3. Moving. The target and the agents can only move horizontally or vertically. At each time step, the target and the agents all make a move simultaneously. Multiple agents are allowed to occupy the same square at any instance in time.
4. Starting position. To create uniformity in the experiments, the target always starts in the middle of the grid, and all the agents are placed beside each other in the lower left corner of the grid. The target is visible to at least one agent. The domain can also be extended to include the starting state where no agent knows the target's initial position (see Section 6).
5. Vision and communication. The agents and target are allowed to "see". Anything that is in an unobstructed direct line from the agent/target (as measured from the center of the source to the center of the destination grid square) is visible to that entity. Furthermore, the agents are allowed to communicate with any agent that is visible to them. This allows an agent to update their private knowledge. Communication occurs between moves and consists of the agents exchanging information as to where they believe the target and the other agents are. No other forms of communication are allowed.
6. Objective. The multiple agents have to catch the target in the fewest possible moves. Catching is defined as being within one square of the target (necessary since the target and agents move at the same time).

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 (A^* variant or alpha-beta) into a framework that, given a goal selection, will find the "best" way to reach the objective. The major problem that an agent faces in this application domain is selecting a search goal. This is complicated by three factors:

1. An agent may not know the location of the target or other agents.

2. The target moves, meaning that the search goal can change from move to move.

3. An agent's local decision should be made in the global context. Agents should take into account what they believe the other agents are doing. Otherwise, many agents may select the same pursuit strategy, reducing the effectiveness of the team.

There are three possible ways for an agent (or the target) to know the location of another agent (or the target):

1. By seeing it (perfect information).

2. Through communication (not applicable to the target).

3. By inference.

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 (opponent modeling – see Section 6), 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. For example, if n time steps have elapsed since the last sighting, the target could be anywhere within n moves from the last known position. Clearly, as n increases, the set of possible target locations can become very large (possibly growing at a rate of n^2). Since the agent cannot explore all potential target locations (the target will probably be long gone by then), the agent must *filter* the set and reduce it to likely locations where it has a chance to find the target.

An agent should choose its destination search area based not only on its beliefs about the target, but also about other agents. Knowing where the other agents are (even if that knowledge is fuzzy) allows an agent to select a search area that (a) is plausible for the target, and (b) reduces/eliminates the possible overlap with what the other agents are doing.

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

Each agent maintains a set of grid locations where the agent believes that the target can be located – the 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 intended to be a real-time search application, and real agents have limited memory, the size of the belief set is limited.

The private knowledge of each agent is updated before each move using any new information obtained by communication or visibility:

1. The agents update their information based on what is visible to them. This may result in them getting a precise location for the target or other agents. As well, all visible empty squares can be eliminated from the belief set.

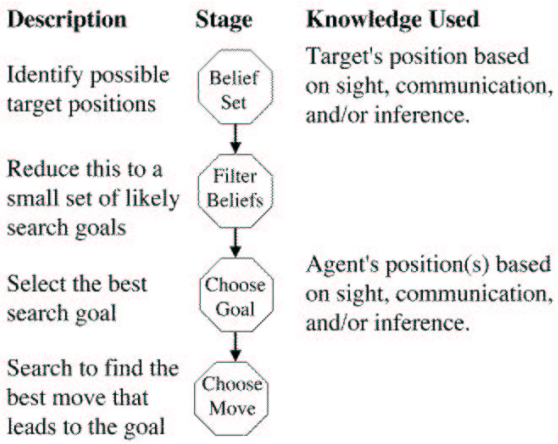

Figure 1: Framework for MAMT solutions

2. The agents are divided into *visibility groups*. A visibility group is a maximal group such that for each pair of agents in the group there is a chain of agents through which the two agents can communicate; in other words, the agents within the group are transitively visible to each other.

3. Within each visibility group, the agent that has the most recent information about an entity (i.e. the target or an agent) conveys that information (and the belief set in the case of the target) to the other agents of the group.

In our work, we implemented three strategies for maintaining the belief set:

1. **All-scenarios.** Expand the current belief set to include all possible locations that can be reached in one more move. If the target has not been seen in n moves, then the belief set includes all grid squares that are reachable in n moves from the last known position (excluding those that can be eliminated by vision).

2. **Region belief set.** This set has the same update as the all-scenarios belief set. However, after the search goal is chosen, the agent *commits* to only consider beliefs in the region of the goal. The belief set may be broken up into connected components (because of visibility), and the agent removes all nodes from the belief set that are not in the connected component containing the search goal.

3. **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.

Note that it is possible for a belief set to become empty. For both the all-scenarios and region belief sets, the location where the target was last spotted becomes the new belief set. When using a single-location strategy, the belief set is reinitialized to be the closest location to the former belief that can't be seen by the agent.

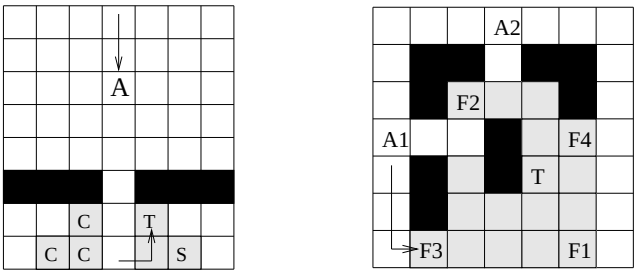

Figure 2: Belief sets (left) and filtering (right)

Figure 2 (left) illustrates the different belief sets. The agent (A) sees the target (T) and both then make two moves. The all-scenarios belief set is represented by the shaded squares. Since the belief set decomposes into two connected components, using a region strategy, the agent would commit to one of the two regions (the left one, C, in this case). With a single-strategy, the agent commits to one square in the belief set (S).

4.2 Filtering

The belief set has to be reduced to a single goal location. This stage is the first step in that direction – it prepares a small subset of the belief set, the *filtered belief set*, for further consideration.

No filtering is needed in the case of the single-location belief set. We use the following greedy algorithm for filtering the all-scenarios and region belief sets. This algorithm tries to come up with an approximation for the "corners" of the variable shaped belief set:

1. A belief with the maximal x-coordinate is the first element of the filtered belief set. The only distinctive property of this belief is that it is located on the border of the belief set.

2. While the number of elements in the filtered belief set is smaller than a predefined constant (4 for our work), do:
   Let S be the belief set and S_1 be the filtered belief set so far. The next element in S_1 is the belief $s \in S$ that maximizes $\min_{b \in S_1} manhattan(s, b)$.

In Figure 2 (right), the squares marked F1, F2, F3 and F4 in order are selected as approximations for the "corners" of the belief set bounding box for agent A1.

4.3 Goal Selection

In addition to the belief set of the target's location, each agent maintains information on the last known location of the other agents. This information is needed to help the agent select a search goal that reduces the probability of duplicating what the other agents are doing.

Each agent selects one member of their filtered belief set to be their search goal. Randomly selecting a location from the filtered belief 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:

1. The distance from the agent to the belief.
2. The minimum of the distances from the belief to the last known positions of the other agents.

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. In Figure 2 (right), agent A1 is closest to F2. A1's knowledge of A2's position suggests that A2 is likely to go towards F2. Hence, A1 chooses to move towards F3 (the closest belief that is farthest away from the other agents).

4.4 Search

Now that each agent has selected a goal, it then 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 (i.e., approximating a fixed amount of time per decision). In the search, the manhattan distance is used as the leaf node evaluation function.

We experimented with two search algorithms:

1. Single-agent minimin search [5]: 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.
2. Adversarial search. The search can alternate between moves for the agent (get closer to 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). The search has the effect of selecting the move that minimizes the worst-case scenario.

4.5 Comments

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. 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.

5 Experiments

5.1 The Moving Target

There are a variety of target strategies that can be investigated (such as the simple avoid, meet, random and stationary used in [3, 2]). Instead, we prefer to use a (minimalist) "realistic" model for the target. The target has limited intelligence based on its knowledge of the last known locations of pursuing agents and the locations it has already visited. The following is a rough sketch of the strategy.

The strategy is a weighted combination of four sub-strategies: distance (d), mobility (m), visibility (v), and random (r). First, the target wants to maximize the *distance* from the agents' location (either real or perceived distance). This can be estimated using manhattan distance. Second, the target wants to maximize its *mobility* by preferring a move that leads to more move choices. By looking a small number of moves ahead, the target can identify which paths lead to more choices (open space), and which to fewer (e.g., dead ends). The third consideration is *visibility*. Moves which increase the real or perceived ability of the agents to see the target are less desirable than moves to squares where the target is not visible. Finally, a random factor was introduced to add some variability to the target's behavior.

Each possible target move is evaluated using the following formula: score = $0.4d + 0.35m + 0.15v + 0.5r$. The maximum scoring move is selected. The weights were hand-tuned based on the perceived realism of the target's behavior. Machine-learning could be used to find a better set of weights.

To avoid cycles the target will not go back to locations it has already visited before, if it has other choices. Overall, the strategy produces reasonable target behavior given its limited search horizon (only used in the mobility calculation).

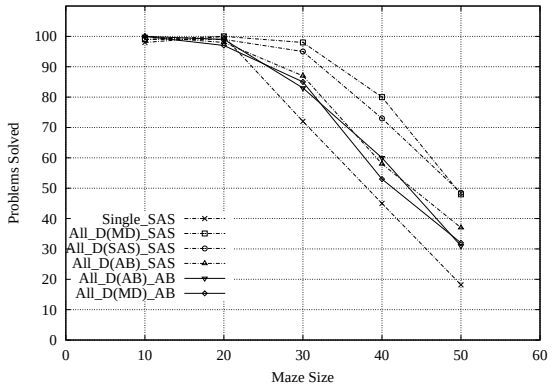

Figure 3: Comparing solutions

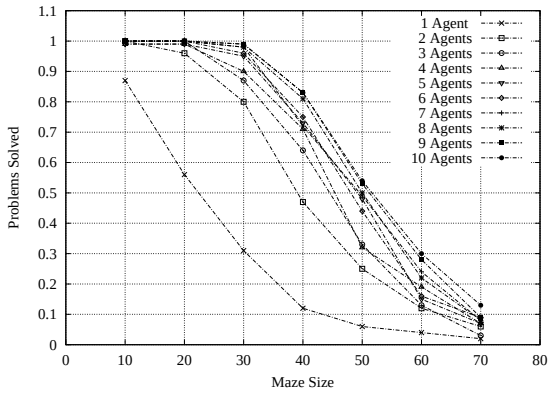

Figure 4: Varying size and # of agents

5.2 Experimental Setup

Solutions to MAMT instances were tested using grids of sizes from 10×10 to 70×70, in increments of 10 for each dimension. Grids had obstacles randomly placed, occupying 10% to 30%, in increments of 5%, of the available space (higher densities result in large parts of the grid becoming unreachable). The experiments ranged from having 1 through 10 agents.

Each pursuer was allocated 50,000 search nodes to make its decision, allowing each step in the pursuit to be completed in less than a second (the worst case is with 10 agents and one target – requiring up to 11 searches per agent per move, i.e. 10 searches for weighting the filtered belief set and 1 search for selecting a move). 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 number of moves was set to $15 \times n$. In our experience, the agents caught the target in less than $8 \times n$ moves, or not at all.

5.3 Experiments

Figure 3 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. For selected solutions, the graph shows the percentage of problems where the target was caught (100 trials per data point) as a function of the maze size.

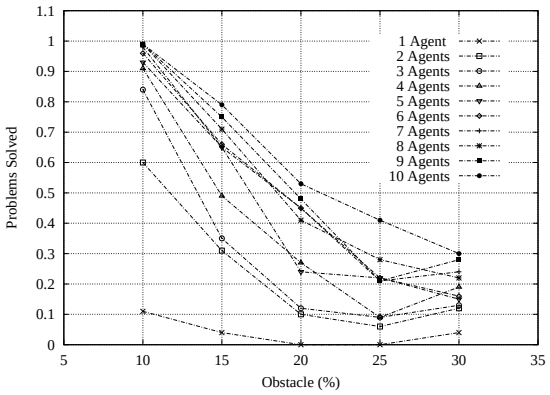

Figure 5: Varying the obstacle density

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 since each agent works in isolation without considering any other agents. 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 investing search in move selection is more beneficial than investing it in goal selection.

The results show that 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, may yield a lesser-quality solution. There are problem instances where alpha-beta's consideration of the target's moves is needed to properly corner the target, but in general these are in the minority.

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. Such an extension is beyond the current scope of our work.

Figure 4 shows that more agents are better than fewer. The version used for this experiment was the one that generally performed best in all our experiments – all-scenarios belief set using the difference metric (manhattan distance) using single-agent search. 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 per search.

Figure 5 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:

1. Multiple moving targets. Why does there have to be only one bad guy?
2. More sophisticated forms of communication can be used (e.g., communicating plans between agents).
3. Agents can use other senses, besides sight. For example, one could enhance the model to include sound. The pursuers could hear a (decaying with distance) sound of running feet that may suggest the target's direction.
4. All agents could have an energy level, that would limit the amount of running they did. When agents don't know what to do, they rest to restore their energy. When they see the target, they sprint to try and catch it.
5. Creating realistic "human-like" target behavior. The target used in this work can be improved on.

The framework used of this research has many opportunities for interesting ideas:

1. Agents can use more sophisticated reasoning about their belief set.
2. Right now, all agents have a complete map of the terrain. A more interesting scenario is to have the agents (pursuers and target) dynamically map the environment. The map could consist entirely of unknown regions. As an agent moves, it can add everything it sees to its map.
3. In our application, initially the target is visible to at least one agent. In the *Minority Report* example, this was not the case. With no information on the target available, the agents should adopt a systematic search pattern.
4. Our current implementation has all states in the belief set being equally likely outcomes. A weighted belief set might yield better results against a predictable opponent (opponent modeling). For example, the belief set could capture the idea of *momentum*; non-random targets tend to continue moving in the same direction. Alternatively, the pursuers could use *anticipation* as part of their decision-making process (e.g., [6]).

5. In the real world, one of the agents is explicitly or implicitly the leader. Typically they issue directions to the other agents that are in ear shot.

The framework given in this paper provides a good starting point for solving problems of having multiple agents pursuing a moving target. There are many ideas yet to be explored in this framework. However, for the purposes of creating "realistic" pursuit scenarios, in particular for real-time-strategy commercial games, the current work has proven to be quite effective.

7 Acknowledgements

This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), Institute for Robotics and Intelligent Systems, and Alberta's Informatics Circle of Research Excellence (iCORE).

References

[1] R. Hill, C. Han, and M. van Lent. Applying perceptually driven cognitive mapping to virtual urban environments. *AAAI*, pages 886–893, 2002.

[2] T. Ishida. Real-time search for autonomous agents and multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 1:139–167, 1998.

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

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

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

[6] J. Laird. Using a computer game to develop advanced AI. *IEEE Computer*, 34(7):70–75, 2001.

[7] J. Reynolds. *Tactical Team AI Using a Command Hierarchy*, pages 260–271. Charles River Media, Inc., 2002.

[8] M. Roth, D. Vail, and M. Veloso. A world model of multi-robot teams with communication, 2003. `www-2.cs.cmu.edu/~mmv/papers/03icra-maayan.pdf`.

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

[10] B. Stout. Smart moves: Intelligent path-finding. *Game Developer Magazine*, pages 28–35, 1996.

[11] W van der Sterren. *Squad Tactics: Planned Maneuvers*, pages 247–259. Charles River Media, Inc., 2002.