Real-time Search in Game-like Environments

Vadim Bulitko

http://ircl.cs.ualberta.ca
Disclaimer

- Work in progress

- Questions/suggestions/criticisms are most welcome
Acknowledgments

- Nathan Sturtevant
- IRCL members
- Adriana Lopez
- Mark Brockington
- Valeriy Bulitko
Outline

- Past work - path refinement in real-time search
- Current work - real-time opponent modeling
- Future work - marriage of the two
Past Work

Path Refinement in Learning Real-time Search
Simultaneous Planning & Learning

- We are interested in domains that are:
  - initially unknown
  - “real-time” (thinking matters)

- Examples:
  - RTS games
  - Robotics
Motivation: Theory

- Sustained attention to decision-making in dynamic, *a priori* unknown domains
- IJCAI’05, AAAI’06, ECAI’06

Challenges to AI

- large scale learning
- planning with uncertainty
- opponent modeling
- interleaving execution and deliberation
Problem Formulation

- path-planning in a game-like environment
- 2D grid
- unknown map
- static goal/start of known coordinates
- trial based
Incremental A*

**Algorithm:**
- run A* with the free space assumption
  - Euclidean distance as the heuristic
- replan when running into an obstacle

**Pros:**
- easy to implement

**Cons:**
- substantial delays when planning
First Move Delay

- Small map from an RPG
- 6,176 states
- Shortest path of 115
- Incremental A*
- several orders of magnitude slower than greedy agent
- Important with many units
- 60-70% of time in AoE2
Search in Abstract Space

Instead of searching in the large ground-level space...

...search in a simpler / smaller / restricted space
Clique Abstraction

Group 1

Group 2

Group 4

Group 3

Figure 30: The process of abstracting a graph from a task domain. We only pay this cost when we are making changes that affect the connectivity of the entire map. In practice, we only pay this cost when we are making changes that affect the connectivity of the entire map. In practice, we only pay this cost when we are making changes that affect the connectivity of the entire map.
Map from an RTS game
Level 0

16,807 nodes
Level 1

- 5,212 nodes
Level 2

- 1,919 nodes
Level 3

- 771 nodes
Clique Abstraction
Path Refinement A*

- **Algorithm:**
  - build/repair clique based abstraction(s)
  - build an abstract path (A*)
  - refine it into a lower level path (A*)

- **Pros:**
  - an implementation is available

- **Cons:**
  - complex

[Sturtevant, Buro, 2005]
Path Refinement
Present Work

Real-time Opponent Modeling in Moving Target Pursuit
NPC following PC

Falling behind
Attacking NPC
Trailing instead of intercepting
Motivation: Applications

- Games
- Law enforcement
- Military
Problem Formulation

- 2D grid
- Target’s position is known frequently
- Target is possibly faster
  - outsmarting vs. outrunning
- Asynchronous environment
  - target moves while you think
  - deliberation time matters
Simulator Cycle

- Each agent:
  - sense
  - deliberate
  - execute
  - wait until next allowed execution time

Think time \((a \times \alpha)\)

Move time \((b)\)

Current time

Next allowed execution time
Moving Target Pursuit

- Trial based chases
- Fixed start position for the target agent
- Fixed start position for the pursuit agent
- Trial ends when:
  - pursuit agent is close to the target
  - travel limit is reached
- Once trial ends, a new trial begins
Search Algorithms

- A* [Hart, et al. 1968]
- D* [Stenz 1995]
- D* Lite [Koenig, Likhachev 2002]
- Trailblazer [Chimura, Tokoro 1994]
- aTrailblazer [Sasaki, et al. 1995]
Too much commitment
Replan

A*
Real-time Search

- MTS [Ishida, Korf 1991]
- MTS with commitment [Ishida 1992]
- MTS with deliberation [Ishida 1992]
Problems

- A*
  - too committing and/or too slow per move

- MTS
  - fast per move but trailing, very slow convergence
Why?

- All these algorithms plan to go where the target presently is.
- One needs to go where the target will be.
- Intercept vs. trail.
**Interception Travel**

$h(s_p,s_t)$ is the expected interception travel of pursuit agent presently in state $s_p$ if target agent is presently in state $s_t$.

- **interception travel < physical distance**
  - Case 1: Pursuit agent travels faster than target.

- **interception travel > physical distance**
  - Case 2: Pursuit agent travels slower than target.
Proposed Approach

- Frequency based motion model \( Pr(s_t \rightarrow s'_t) \)

- Real-time dynamic programming [Barto, et al. 1995]

- Bellman equation

\[
h^*(s_p, s_t) = \min_{s'_p} \sum_{s'_t} Pr(s_t \rightarrow s'_t) \left[ c(s_p, s'_p) + h^*(s'_p, s'_t) \right]
\]

- Turned into update rule:

\[
h_{new}(s_p, s_t) \leftarrow (1-\alpha)h_{old}(s_p, s_t) + \alpha \min_{s'_p} \sum_{s'_t} Pr(s_t \rightarrow s'_t) \left[ c(s_p, s'_p) + h_{old}(s'_p, s'_t) \right]
\]

- Epsilon-greedy control policy
Initial, \( \epsilon = 0.3 \)
After < 100 trials
OM MTP vs. MTS

Reduction in convergence travel, level 0, patrolUnit, empty.txt

Thinking penalty
Speed advantage of the pursuit agent

0.5 0.6 0.7 0.8 0.9 1 1.1 1.25 1.43 1.67 2

0
1
5
10
50
100
500
1000
−1
−0.5
0
0.5
1
x 10^4
Thinking penalty
Speed advantage of the pursuit agent
Reduction in convergence travel, level 0, patrolUnit, empty.txt

0.5 0.6 0.7 0.8 0.9 1 1.1 1.25 1.43 1.67 2

0
1
5
10
50
100
500
1000

−1
−0.5
0
0.5
1

x 10^4
Thinking penalty

Speed advantage of the pursuit agent

Reduction in convergence travel, level 0, patrolUnit, empty.txt

0.5 0.6 0.7 0.8 0.9 1 1.1 1.25 1.43 1.67 2

0
1
5
10
50
100
500
1000

$-1$

$-0.5$

0

0.5

1

$10^4$
Thinking penalty
Speed advantage of the pursuit agent
Reduction in convergence travel, level 0, patrolUnit, empty.txt

0.5 0.6 0.7 0.8 0.9 1 1.1 1.25 1.43 1.67 2

0 1 5 10 50 100 500 1000

-1 -0.5 0 0.5 1

x 10^4
Game Maps

- HOG simulator
- clique abstraction
- patrol unit as the target
- a commercial game map, 16,807 passable states
- 50 random problems, initial distance 60-100
- Proximity tolerance: 1 cell
- Target twice as fast
Convergence

- A*
- MTS
- OM MTP

Each point averaged over 50x30 data
Statistics

- 300 trials
- travel limited to 200 on each

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Travel</th>
<th>Final trial</th>
<th>Heuristic Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>43,173 ± 2,372</td>
<td>144 ± 7</td>
<td>0</td>
</tr>
<tr>
<td>MTS</td>
<td>41,282 ± 2,560</td>
<td>132 ± 9</td>
<td>547 ± 69</td>
</tr>
<tr>
<td>OM MTS</td>
<td>38,746 ± 2,359</td>
<td>126 ± 9</td>
<td>1,120 ± 859</td>
</tr>
</tbody>
</table>
Future Work

Real-time Opponent Modeling and Path Refinement in Moving Target Pursuit
Problems with OM MTP

- Slow convergence
- Heuristic is target-position specific
- Quadratic memory requirements (doubled)
A Happy Marriage?

- Opponent modeling + real-time dynamic programming in an abstract state space

- Refine the solution with A* in the corridor
Why Would It Work?

- Learning heuristic (RTDP) and opponent modeling (OM) can work better in an abstract space:
  - smaller size
  - more deliberation per target’s move
  - each value affects several ground states

- $A^*$ as path-refinement engine works due to:
  - applied in a narrow corridor
  - applied for a short distance
Just in:
Related Research at
Summary

- Real-time search: static/moving target
- Problems: slow convergence, slow response, trailing
- Solution:
  - state abstraction + path refinement
  - opponent modeling + real time dynamic programming

http://ircl.cs.ualberta.ca