Real-time
Search & Learning

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

- Work in progress

- Questions/suggestions/criticisms are most welcome
Acknowledgments

Nathan Sturtevant [collaborator]
IRCL members
Adriana Lopez
Mark Brockington
Valeriy Bulitko
Emilie Kohler
Kevin Murphy
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
A
B
C
K
J
Group 2
E
D
Group 3
F
G
Group 4
I

Figure 30 The process of abstracting a map representation. 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
LRTA*

**Algorithm:**
- greedy
- heuristic is updated on every move

**Pros:**
- real-time

**Cons:**
- large memory requirements
- slow, unstable convergence

[Korf, 1990]
LRTA* in action
First move lag:
6 nodes \((484x\ faster)\)

Convergence travel: \(47630\)
\((160x\ slower)\)
RTA*/LRTA* [Korf 90]
MTS [Ishida, Korf 91]
SRTA* [Russell, Wefald 91]
DTA*/LDTA* [Russell, Wefald 91]
SLA* [Shue, Zamani 93]
RTDP [Barto, et al. 95]
Weighted LRTA* [Ishida, Shimbo 96]
Bounded LRTA* [Ishida, Shimbo 96]
RTBS [Ishida, 97]
SLA*T [Shue, Zamani 98]
FALCONS [Furcy, Koenig 00]
eFALCONS [Furcy, Koenig 01]
CB-LRTA* [Shang, et al. 03]
Multi-update LRTA* [Koenig 04]
ϒ-Trap [Bulitko 04]
LRTS [Bulitko, Lee 06]
LRTA*(k) [Hernandez, Meseguer 05]
P-LRTA* [Rayner, et al. 06]
Learning Real-time Search (LRTS)

Algorithm:
- LRTA* extended with max-of-min lookahead, weights and backtracking

Pros:
- faster and more stable convergence
- more memory efficient

Cons:
- more complex
- can still be memory-prohibitive for MTS

[Bulitko, Lee, 2006]
LRTS in action

heuristic update

d
First move lag: 789 nodes (132x slower)
Convergence travel: 1217 (39x faster)
Outstanding Issues

- Slow convergence
- $\Theta$(number of states) memory required
- Why?
  - Large state space
  - Tabular representation of heuristic function
  - No generalization
Generalization

- heuristic function approximation
  - e.g., linear function with tile-coding
- state aggregation
  - tiling, clique abstraction, triangulation
Path refinement LRTS

Algorithm:
- Subsumes incremental A*, PRA*, LRTS
- Arbitrary algorithms for abstraction levels
- Path-refinement

Pros:
- faster convergence, less memory
- lots of flexibility

Cons:
- more complex
- user-defined parameters

[Bulitko, Sturtevant, Kazakevich, 2005]
Path Refinement LRTS

- First move lag: 262 nodes (3x faster)
- Convergence travel: 619 (2x faster)
Evaluation

- 10 maps from a role-playing video game
- 1,000 convergence runs total

- LRTS\((d=10, \gamma=0.5, T=0)\)

- PR LRTS
  - A* at level 0
  - LRTS\((d=5, \gamma=0.5, T=0)\) at level 1
Effects of Abstraction

- PR LRTS over LRTS
- 40% reduction in convergence travel
- the same time lag
- the same suboptimality

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1st move time</th>
<th>Conv. travel</th>
<th>Suboptimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>5.01 ms</td>
<td>186</td>
<td>0.0%</td>
</tr>
<tr>
<td>LRTA*</td>
<td>0.02 ms</td>
<td>25,868</td>
<td>0.0%</td>
</tr>
<tr>
<td>LRTS</td>
<td>0.93 ms</td>
<td>555</td>
<td>2.07%</td>
</tr>
<tr>
<td>PR-LRTS</td>
<td>0.95 ms</td>
<td>345</td>
<td>2.19%</td>
</tr>
</tbody>
</table>
Why Does it Work?

- At higher abstraction levels
  - A* spends less time planning
  - LRTS has fewer heuristic values to learn
- Each heuristic value learnt at an abstract level affects several ground states
- Search at the lower level is restricted
Chasing

[Copyright 1999 Warner Bros.]
Present Work

Real-time Opponent Modeling in Moving Target Pursuit
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
Motivation: Applications

- Games

- Law enforcement

- Military
Falling behind

Game AI
Game AI

Trailing instead of intercepting
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
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*

Trailing, slow
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
Proposed Approach

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

$$h(s_p, s_t) \leq c(s_p, s'_p) + h(s'_p, s'_t)$$
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
Model

simulator stochasticity
Initial, \( \epsilon = 0.3 \)
After < 100 trials
Preliminary Experiments

- 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
Even More Future Work

- Opponent model
- that takes our move into account: \( Pr(s_t \rightarrow s'_t | s_p \rightarrow s'_p) \)
- function approximation for target motion models [Paduraru 2006]
- predictive state representations for target motion models
- Function approximation for value function (e.g., tile coding, neural networks, etc.)
- Dynamic environments
- Actively evasive opponents (e.g., mini-max)
- Tactical formations
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