Speeding Up Learning via Abstraction

Vadim Bulitko
(joint work with Nathan Sturtevant & Maryia Kazakevich)
University of Alberta
http://www.cs.ualberta.ca/lrts
Outline

× Motivation
× Incremental Search
× Learning Real-time Search
× State Abstraction
× Speed Up
Motivation

Some domains can be:

- Initially unknown
- “Real-time” (thinking time matters)
- Dynamic
- Stochastic

Examples:

- RTS games
- Robotics
- Multi-agent systems
- Decision-support systems
Testbed Used

- Hierarchical Open Graph
- Commercial maps
  - Baldur’s Gate
  - Warcraft 3
- Initially unknown map
- Multi-agent path-planning
- A number of algorithms implemented already
Problem Formulation

- Two-dimensional 8-connected grid
- Static start, goal states of known coordinates
  - Octile distance as the initial heuristic
- Map is initially unknown
- The agent has a visibility radius of 10
- Each action costs: 1 or $\sqrt{2}$
- **Trial**: a sequence of actions from start to goal
- **Convergence run**: a sequence of trials until the agent behavior is guaranteed to stabilize (e.g., no updates to the heuristic function)
Measurements

- Convergence travel
- First-move lag on the last trial
- Final solution suboptimal
Example

start state

goal state
Questions Addressed

✗ How can the first-move delay (lag) be minimized?

✗ How can the agent make local decisions and yet come up with a reasonable global solution?

✗ Given repeated experiences, how can learning be accelerated to minimize the number of trials?
Cannot run A*/IDA* directly since the map is unknown

Simple solution:
- Freespace assumption
- Replan when needed

Incremental A*

Demo (‘x’)

Can re-use the information between re-planning episodes
- Dynamic A* (D*) [Stenz, 95]
- D* Lite [Koenig/Likhachev, 02]
- Doesn’t help with the first-move lag
First-move Lag

× Small map from Baldur’s Gate game
× 6,176 states
× Start: (31,62)
× Goal: (91,120)
× Shortest path length: 115.4
× Incremental A*: 
  - First-move lag: 17 ms
  - 1,900 times slower than LRTA* (d=1)
× Also, see [Koenig, 04]
Real-time Search

× Pioneered by Korf [Korf, 90]
× LRTA*
  ∙ Full width, limited depth lookahead
  ∙ Best frontier node \( x \) with the smallest \( f = g + h \)
  ∙ \( h(\text{current}) = f(\text{best frontier node}) \)
  ∙ Takes a step towards \( x \)
× Complete & Optimal (w/ fine print)
× Beautiful in its simplicity! (w/o fine print)
A considerable following...

- RTA*/LRTA* [Korf 1990]
- MTS [Ishida, Korf 1991]
- SRTA* [Russell, Wefald 1991]
- DTA*/LDTA* [Russell, Wefald 1991]
- SLA* [Shue, Zamani 1993]
- RTDP [Barto, et al. 1995]
- Weighted LRTA* [Ishida, Shimbo 1996]
- Bounded LRTA* [Ishida, Shimbo 1996]
- RTBS [Ishida, 1997]
- SLA*T [Shue, Zamani 1998]
- FALCONS [Furcy, Koenig 2000]
- eFALCONS [Furcy, Koenig 2001]
- CB-LRTA* [Shang, et al. 2003]
- Multi-update LRTA* [Koenig 2004]
- γ-Trap [Bulitko 2004]
- LRTS [Bulitko 2004]
Three extensions

- Deeper lookahead
  - [Korf 1990]
- Weighted g-distance
  - [Shimbo/Ishida 2003]
- Learning quota, backtracking

- Learning Real-time Search (LRTS)
- LRTS(d,\(\gamma\),T)  [Bulitko 2004]
Overall Impact

✗ Can have:
- 50x faster convergence than LRTA*
- 5x shorter first-move lag than incremental A*
- At the cost of suboptimality (~ 2%)

<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>
</tbody>
</table>
Problems Left

- Even on small maps hundreds of moves needed for convergence
- Non-monotonic convergence
- Demo (‘u’, 77 trials)
Low-level Search

✗ You are planning a road trip from Los Angeles to Edmonton (~3,400km)
✗ Would you look at a map of New York?

✗ No, we seem to derive a high-level plan first and \textbf{refine} it:
  \begin{itemize}
    \item Go north
    \item Take I-5
    \item Take Figuero Street onto I-110
    \item ...
  \end{itemize}
No Generalization in ML

× Heuristic is represented in tabular form
× Learnt one value per move
× No generalization of updates

× Humans and animals appear to generalize their learning experiences
  - From one situation
  - Onto a set of similar situations
How To Generalize?

Several methods possible, including:
- Heuristic function approximation
- State abstraction

We try the latter in this work

More thoughts are at:
- http://www.cs.ualberta.ca/lrts
Building State Abstraction
An Example
Repair Demo (‘x’)

February 2005
Costs

× To build: $O(V)$
  - $V$ is the size of the map

× To repair:
  To remove an edge:
    - $O(1)$ (usual)
    - $O(\log n)$ (worst)
  - $n$ is the size of the abstraction level
Hand Trace

Level 1
LRTS(d=2)

Level 0
A*
× Populate levels 0 to N with either an A* or LRTS
× Create an abstract path $p_N$ at level $N$
× Widen it
× Refine it to level $N-1$ : $c_{N-1}$
× Create an abstract path $p_{N-1}$ by reasoning within the corridor $c_{N-1}$
× Repeat until level 0

× Path-refinement learning real-time search (PR LRTS)
Demo ( ‘f’ )
Empirical Evaluation (1)

× PR LRTS over LRTS:
  • 40% reduction in convergence travel
  • About the same time lag
  • About 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>
Empirical Evaluation (2)

![Graph showing the comparison of different algorithms (A*, LRTA*, LRTS, and PR-LRTS) in terms of first move lag vs. solution length.](image-url)
Empirical Evaluation (3)

Convergence Travel

x 10^4

Solution Length

0 10 20 30 40 50 60 70 80

0 1 2 3 4

February 2005 © Vadim Bulitko: University of Alberta
Empirical Evaluation (4)
Why Does it Work?

- Higher abstraction levels --> smaller state space
  - A* spends less time
  - LRTS has fewer heuristic values to learn

- Each heuristic value learnt at a higher abstraction level affects several/many ground states. Generalization.

- Algorithms operating at lower levels are restricted by a “corridor” from the top level
Future Work

- Selection of abstraction levels
- Reduction in memory
  - Applications to MTS
- Multi-agent environment
  - Co-operative
  - Non co-operative
- Abstraction in other domains
Summary

- Investigated learning real-time search applied to real-time path-finding
- Showed influence of three extensions to LRTA*
- Extended LRTS with dynamically built/repaired state abstraction
- Showed benefits gained
Thanks

http://www.cs.ualberta.ca/lrts
Extension 1

× Proposed by Korf in the original paper
× Deeper lookahead (depth d)
  ◦ Some care of consistency ("path-max")
× Can take only one step
× Can take d steps

<table>
<thead>
<tr>
<th>d</th>
<th>Deliberation per move (ms)</th>
<th>Travel per trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0087</td>
<td>661.5</td>
</tr>
<tr>
<td>3</td>
<td>0.0215</td>
<td>241.8</td>
</tr>
<tr>
<td>5</td>
<td>0.0360</td>
<td>193.3</td>
</tr>
<tr>
<td>7</td>
<td>0.0514</td>
<td>114.9</td>
</tr>
<tr>
<td>9</td>
<td>0.0715</td>
<td>105.8</td>
</tr>
</tbody>
</table>
Extension 2

- Proposed by [Shimbo/Ishida 2003]
- Scale the initial heuristic by 1+epsilon
- Equivalent to weighting the g-distance by 0<gamma<=1 [Bulitko, 2004]
- Faster convergence
- Suboptimal (up to 1/gamma) solutions
- Similar effects seen with A* [Korf 1993]

<table>
<thead>
<tr>
<th>γ</th>
<th>Suboptimality</th>
<th>Convergence travel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>6.19%</td>
<td>9,300</td>
</tr>
<tr>
<td>0.3</td>
<td>4.92%</td>
<td>8,751</td>
</tr>
<tr>
<td>0.5</td>
<td>2.41%</td>
<td>9,435</td>
</tr>
<tr>
<td>0.7</td>
<td>1.23%</td>
<td>13,862</td>
</tr>
<tr>
<td>0.9</td>
<td>0.20%</td>
<td>25,507</td>
</tr>
<tr>
<td>1.0</td>
<td>0.00%</td>
<td>31,336</td>
</tr>
</tbody>
</table>
Extension 3

- Proposed in [Shue/Zamani 1993]
- Go to the previous state when h(current) is updated
- Results in faster convergence
- Longer first trial

<table>
<thead>
<tr>
<th>$T$</th>
<th>First trial travel</th>
<th>Convergence travel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>434</td>
<td>457</td>
</tr>
<tr>
<td>10</td>
<td>413</td>
<td>487</td>
</tr>
<tr>
<td>50</td>
<td>398</td>
<td>592</td>
</tr>
<tr>
<td>1,000</td>
<td>390</td>
<td>810</td>
</tr>
<tr>
<td>5,000</td>
<td>235</td>
<td>935</td>
</tr>
</tbody>
</table>
Repairing State Abstraction