Recursive Best-First Search with Bounded Overhead

Matthew Hatem and Scott Kiesel and Wheeler Ruml

University of New Hampshire

with support from NSF grant IIS-1150068
A* remembers every state it visits, often exceeds memory
A* remembers every state it visits, often exceeds memory

Motivation for linear-space variants:
A* remembers every state it visits, often exceeds memory

Motivation for linear-space variants:

- Iterative Deepening A* (Korf 1985)
**Problem Summary**

- A* remembers every state it visits, often exceeds memory
- Motivation for linear-space variants:
  - Iterative Deepening A* (Korf 1985)
  - Has bounded overhead
A* remembers every state it visits, often exceeds memory

Motivation for linear-space variants:

- Iterative Deepening A* (Korf 1985)
  - Has bounded overhead
  - Only best-first in some cases!
A* remembers every state it visits, often exceeds memory.

Motivation for linear-space variants:

- Iterative Deepening A* (Korf 1985)
  - Has bounded overhead
  - Only best-first in some cases!
- Recursive Best-First Search (Korf 1993)
A* remembers every state it visits, often exceeds memory

Motivation for linear-space variants:

- **Iterative Deepening A***(Korf 1985)**
  - Has bounded overhead
  - Only best-first in some cases!

- **Recursive Best-First Search (Korf 1993)**
  - Always best-first!
A* remembers every state it visits, often exceeds memory.

Motivation for linear-space variants:

- Iterative Deepening A* (Korf 1985)
  - Has bounded overhead
  - Only best-first in some cases!

- Recursive Best-First Search (Korf 1993)
  - Always best-first!
  - Suffers from thrashing overhead
Problem Summary

- A* remembers every state it visits, often exceeds memory

Motivation for linear-space variants:

- Iterative Deepening A* (Korf 1985)
  - Has bounded overhead
  - Only best-first in some cases!

- Recursive Best-First Search (Korf 1993)
  - Always best-first!
  - Suffers from thrashing overhead
    - This is what we fix!
Iterative Deepening Depth-First Search (IDDFS)

- Problem
- IDDFS
- IDA*
- RBFS
- RBFS_{CR}

Depth Bound = 0
Iterative Deepening Depth-First Search (IDDFS)
Iterative Deepening Depth-First Search (IDDFS)

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS\textsubscript{CR}

Evaluation

Conclusion

Depth Bound = 2
Iterative Deepening Depth-First Search (IDDFS)

Depth Bound = 3
Iterative Deepening Depth-First Search (IDDFS)

Depth Bound = 4
Iterative Deepening Depth-First Search (IDDFS)
Iterative Deepening Depth-First Search (IDDFS)

Depth Bound = 6
Iterative Deepening A* (IDA*)

Cost Bound = 6

\[ f(n) = g(n) + h(n) \]

"f layers"
Iterative Deepening A* (IDA*)

**Background**

- Problem
- IDDFS
- IDA*
- RBFS
- RBFS_{CR}

**Evaluation**

**Conclusion**

Cost Bound = 6

\[ f(n) = g(n) + h(n) \]

"f layers"

Only best-first if \( f \) layers are monotonically increasing!

(not the case in, eg, suboptimal variants)
# Recursive Best-First Search (RBFS)

## Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS_CR

## Evaluation

## Conclusion
Recursive Best-First Search (RBFS)

- Problem
- IDDFS
- IDA*
- RBFS
- RBFS_CR

Evaluation

Conclusion

- best 5
- next best 7
- bound = 7
Recursive Best-First Search (RBFS)

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS\_CR

Evaluation

Conclusion
Recursive Best-First Search (RBFS)

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS
- CR

Evaluation

Conclusion

Wheeler Ruml (UNH)
Recursive Best-First Search (RBFS)

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS_C

Evaluation

Conclusion
Recursive Best-First Search (RBFS)
Recursive Best-First Search (RBFS)

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS\text{CR}

Evaluation

Conclusion
Recursive Best-First Search (RBFS)
Strict best-first search order causes thrashing!
RBFS with Bounded Overhead

Background
- Problem
- IDDFS
- IDA*
- RBFS

Evaluation

Conclusion

\[ \text{min} + \text{epsilon} \]

min + epsilon
RBFS with Bounded Overhead

Background
- Problem
- IDDFS
- IDA*
- RBFS
- RBFS CR

Evaluation

Conclusion

Wheeler Ruml (UNH)
Relaxing best-first search order reduces overhead! 
See paper for proof
Korf 100 (Korf 1985)

A* with Manhattan Distance runs out of memory
Korfs 100 15 puzzles (unit cost)

<table>
<thead>
<tr>
<th>Method</th>
<th>Exp. Cost</th>
<th>Avg. Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBFS</td>
<td>29,253,944</td>
<td>62</td>
<td>44</td>
</tr>
<tr>
<td>RBFS$_\epsilon$</td>
<td>13,078,227</td>
<td>71</td>
<td>30</td>
</tr>
<tr>
<td>RBFS$_{CR}$</td>
<td>11,695,743</td>
<td>66</td>
<td>56</td>
</tr>
<tr>
<td>IDA*</td>
<td><strong>11,136,196</strong></td>
<td>68</td>
<td><strong>11</strong></td>
</tr>
</tbody>
</table>

Relaxing best-first search order gives faster solving times
Better solutions but slower than IDA*
From Ghallab, Nau, Traverso (2004)
- All actions have real costs
- Many duplicate states
- $\text{IDA}^*_{CR}$ and $\text{RBFS}_{CR}$ with transposition tables
5 locations, cranes, piles and 8 containers

<table>
<thead>
<tr>
<th></th>
<th>Exp.</th>
<th>Exp./Sec.</th>
<th>Time</th>
<th>Reopened</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDA*&lt;sub&gt;CR&lt;/sub&gt;</td>
<td>2,044m</td>
<td>112k</td>
<td>18,394</td>
<td>2,042m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;CR&lt;/sub&gt;</td>
<td>188m</td>
<td>103k</td>
<td>1,824</td>
<td>87m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=1&lt;/sub&gt;</td>
<td>472m</td>
<td>147k</td>
<td>3,222</td>
<td>4m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=2&lt;/sub&gt;</td>
<td>251m</td>
<td>140k</td>
<td>1,795</td>
<td>5m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=3&lt;/sub&gt;</td>
<td>154m</td>
<td>141k</td>
<td>1,094</td>
<td>14m</td>
</tr>
</tbody>
</table>
Dockyard Robot Planning

5 locations, cranes, piles and 8 containers

<table>
<thead>
<tr>
<th>Method</th>
<th>Exp.</th>
<th>Exp./Sec.</th>
<th>Time</th>
<th>Reopened</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDA*&lt;sub&gt;CR&lt;/sub&gt;</td>
<td>2,044m</td>
<td>112k</td>
<td>18,394</td>
<td>2,042m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;CR&lt;/sub&gt;</td>
<td>188m</td>
<td>103k</td>
<td>1,824</td>
<td>87m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=1&lt;/sub&gt;</td>
<td>472m</td>
<td>147k</td>
<td>3,222</td>
<td>4m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=2&lt;/sub&gt;</td>
<td>251m</td>
<td>140k</td>
<td>1,795</td>
<td>5m</td>
</tr>
<tr>
<td>RBFS&lt;sub&gt;ε=3&lt;/sub&gt;</td>
<td>154m</td>
<td>141k</td>
<td>1,094</td>
<td>14m</td>
</tr>
</tbody>
</table>

RBFS<sub>CR</sub> is faster because it expands 10x fewer nodes
IDA* is depth-first, RBFS is always best-first!
IDA* is depth-first, RBFS is always best-first!

Simple modifications reduce overhead (see paper)
- IDA* is depth-first, RBFS is always best-first!
- Simple modifications reduce overhead (see paper)
- \( RBFS_{CR} \) gives first provable bounds for overhead
- IDA* is depth-first, RBFS is always best-first!
- Simple modifications reduce overhead (see paper)
- $\text{RBFS}_{\text{CR}}$ gives first provable bounds for overhead
- Is it time for IDA* to retire?
Summary

- IDA* is depth-first, RBFS is always best-first!
- Simple modifications reduce overhead (see paper)
- RBFS\textsubscript{CR} gives first provable bounds for overhead
- Is it time for IDA* to retire?
- RBFS deserves more attention!
  - IDA* citations: 1523
  - RBFS citations: 310
15-Puzzle (unit w=4.0)

- **RBFS-cr**
- **RBFS**
Anytime Heuristic Search

15-Puzzle (sqrt w=4.0)

Solution Quality

CPU Time

RBFS-cr

RBFS
25 Pancake (sqrt w=1.2)