Parallel Best-First Search: Optimal and Suboptimal Solutions

Ethan Burns, Seth Lemons, Wheeler Ruml

Supported in part by NSF grant IIS-0812141.

Rong Zhou

University of New Hampshire

Palo Alto Research Center
Now we’re into the explicit parallelism multiprocessor era, and this will dominate for the foreseeable future. I don’t see any technology or architectural innovation on the horizon that might be competitive with this approach.

*John Hennessy*

President of Stanford University,
Cofounder of MIPS Computer Systems

*(A Conversation with John Hennessy and David Patterson, ACM Queue, December 2006)*
Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)

- Parallelized breadth-first search.
- Used abstraction to divide labor between threads.
Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)

- Parallelized breadth-first search.
- Used abstraction to divide labor between threads.

New: Parallel Best $N$-Block First Search

- Approximates best-first ordering.
- Requires care to avoid livelock.
- Broadly applicable framework.
■ Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)
  ◆ Parallelized breadth-first search.
  ◆ Used abstraction to divide labor between threads.

■ New: Parallel Best $N$Block First Search
  ◆ Approximates best-first ordering.
  ◆ Requires care to avoid livelock.
  ◆ Broadly applicable framework.

■ New: Two Pruning Rules for Parallel Suboptimal Search
  ◆ Pruning against an incumbent.
  ◆ Pruning some duplicates.
Naive Parallel Search

Introduction
- Motivation
- Overview
- Parallel Search
- Abstraction

PBNF

Suboptimal Search

Anytime Search

Conclusion
Naive Parallel Search

Introduction
- Motivation
- Overview
- Parallel Search
- Abstraction

Suboptimal Search
- PBNF

Anytime Search

Conclusion
Naive Parallel Search

Introduction
- Motivation
- Overview
- Parallel Search
- Abstraction

PBNF

Suboptimal Search

Anytime Search

Conclusion
Work is divided among threads using a special hash function based on abstraction.

- Few possible locations for children.
Work is divided among threads using a special hash function based on abstraction.

- Threads acquire groups of states called *n* blocks.
- Exclusive access to a duplicate detection scope.
Work is divided among threads using a special hash function based on abstraction.

- Disjoint duplicate detection scopes searched in parallel.
New: Parallel Best \( N \) Block First Search
1. Search disjoint $n$ blocks in parallel.
   - Maintain a heap of free $n$ blocks.
   - Greedily acquire best free $n$ block (and its scope).
   - Each $n$ block is searched in $f(n)$ order.
1. Search disjoint $n$ blocks in parallel.
   - Maintain a heap of free $n$ blocks.
   - Greedily acquire best free $n$ block (and its scope).
   - Each $n$ block is searched in $f(n)$ order.

2. Switch $n$ blocks when a better one becomes free.
   - Approximates best-first expansion order.
   - Perform a minimum amount of work before switching.
Parallel Best $N$ Block First Search

1. Search disjoint $n$ blocks in parallel.
   - Maintain a *heap of free $n$ blocks*.
   - Greedily acquire best free $n$ block (and its scope).
   - Each $n$ block is searched in $f(n)$ order.

2. Switch $n$ blocks when a better one becomes free.
   - Approximates best-first expansion order.
   - Perform a minimum amount of work before switching.

3. Stop when the incumbent solution is optimal.
   - Prune nodes on the cost of the incumbent
   - Incumbent is optimal when all nodes are pruned.
Problem with best free block ordering:
Problem with best free \( n \) block ordering:
Problem with best free block ordering:
Problem with best free block ordering:
■ No guarantee that a given \textit{n}block will become free.
  
  ◆ In infinite search spaces, there can be livelock.

■ Solution: check for \textit{hot} \textit{n}blocks
  
  ◆ Flag better \textit{n}blocks as \textit{hot}
  ◆ Release an \textit{n}block to free an interfered hot \textit{n}block.
Solution:
Solution:
Solution:
## Solution:

![Diagram of a network with a highlighted path]

- Safe PBNF
- Introduction
- PBNF
- Livellock
- Empirical Evaluation
- Planning
- Summary
- Suboptimal Search
- Anytime Search
- Conclusion
Solution:
Empirical Evaluation

Software

- C++, POSIX threads, jemalloc library
- Fedora 9

Hardware

- Dual quad-core Intel Xeon E5320 1.86GHz 64-bits
- 16Gb RAM

Domains

- Grid pathfinding
  - Abstraction: courser grid
- 15 puzzles
  - Abstraction: ignore some tile numbers
- STRIPS planning
  - Abstraction: generated automatically
**PA***

- Basic A* with a lock on open and closed lists.

**Lock-free PA***

- PA* with lock-free data structures.

**KBFS** (Felner et al., 2003)

- Expand the $K$ best open nodes in parallel.

**PRA*** (Evett et al., 1995)

- Hash nodes to distribute among processors.
- Synchronized message queues for “incoming” nodes.

**IDPSDD**

- PSDD with iterative-deepening for bounds.
Four-way Grid Pathfinding (Previous Algorithms)

- Introduction
- PBNF
  - Livellok
  - Empirical Evaluation
- Planning
- Summary
- Suboptimal Search
- Anytime Search
- Conclusion

Grid Unit 4-Way (Previous Algorithms)

![Graph showing wall time (seconds) vs. threads for different algorithms including Lock-free PA*, KBFS, PA*, PSDD, PRA*, and Serial A*.

Seth Lemons (UNH) Parallel Best-First Search: Optimal and Suboptimal Solutions – 11 / 39
APRA*

- PRA* with a novel abstraction based hashing.

PSDD (Zhou and Hansen, 2007)

- Abstraction to find disjoint portions of a search space.
- Breadth-first search
- All threads synchronize at each layer

BFPSDD

- PSDD with \( f(n) \) layers instead of depth layers.

(Safe) PBNF

- Acquire the best free \( n \)block (prevent live-locks).
Four-way Grid Pathfinding (Optimal)

Introduction

PBNF

- Livelong
- Empirical Evaluation
- Planning
- Summary

Suboptimal Search

Anytime Search

Conclusion

Grid Unit 4-Way

wall time (seconds)

threads

Serial A*
APRA*
BFPSSD
SafePBNF
PBNF
Easy Sliding 15-Puzzles (Optimal)

15-Puzzles

wall time (seconds)

threads

IDPSDD
BFPSDD
Serial A*
APRA*
SafePBNF
PBNF

Seth Lemons (UNH) Parallel Best-First Search: Optimal and Suboptimal Solutions – 14 / 39
### A* Comparison

<table>
<thead>
<tr>
<th></th>
<th>threads</th>
<th>logistics-6</th>
<th>blocks-14</th>
<th>gripper-7</th>
<th>satellite-6</th>
<th>elevator-12</th>
<th>freecell-3</th>
<th>depots-7</th>
<th>driverlog-11</th>
<th>gripper-8</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>5.2</td>
<td>118</td>
<td>131</td>
<td>336</td>
<td>199</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>528</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>7.1</td>
<td>60</td>
<td>96</td>
<td>213</td>
<td>150</td>
<td>301</td>
<td>322</td>
<td>528</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.76</td>
<td>5.5</td>
<td>51</td>
<td>49</td>
<td>269</td>
<td>112</td>
<td>144</td>
<td>103</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.2</td>
<td>3.8</td>
<td>41</td>
<td>66</td>
<td>241</td>
<td>61</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.84</td>
<td>3.7</td>
<td>28</td>
<td>49</td>
<td>169</td>
<td>40</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

### Experimental Results

**APRA**

<table>
<thead>
<tr>
<th></th>
<th>threads</th>
<th>logistics-6</th>
<th>blocks-14</th>
<th>gripper-7</th>
<th>satellite-6</th>
<th>elevator-12</th>
<th>freecell-3</th>
<th>depots-7</th>
<th>driverlog-11</th>
<th>gripper-8</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3</td>
<td>6.3</td>
<td>40</td>
<td>68</td>
<td>157</td>
<td>186</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.72</td>
<td>3.8</td>
<td>16</td>
<td>34</td>
<td>56</td>
<td>64</td>
<td>M</td>
<td>M</td>
<td>96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.58</td>
<td>2.7</td>
<td>11</td>
<td>21</td>
<td>35</td>
<td>44</td>
<td>M</td>
<td>M</td>
<td>61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.53</td>
<td>2.6</td>
<td><strong>8.6</strong></td>
<td><strong>17</strong></td>
<td><strong>27</strong></td>
<td><strong>36</strong></td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>48</td>
<td></td>
</tr>
</tbody>
</table>

**PNBF**

<table>
<thead>
<tr>
<th></th>
<th>threads</th>
<th>logistics-6</th>
<th>blocks-14</th>
<th>gripper-7</th>
<th>satellite-6</th>
<th>elevator-12</th>
<th>freecell-3</th>
<th>depots-7</th>
<th>driverlog-11</th>
<th>gripper-8</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>6.2</td>
<td>40</td>
<td>77</td>
<td>150</td>
<td>127</td>
<td>156</td>
<td>154</td>
<td>235</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.64</td>
<td>2.7</td>
<td>17</td>
<td>24</td>
<td>54</td>
<td>47</td>
<td>63</td>
<td>60</td>
<td>98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.56</td>
<td>2.2</td>
<td>11</td>
<td>17</td>
<td>34</td>
<td>38</td>
<td>43</td>
<td>39</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.62</td>
<td><strong>2.0</strong></td>
<td><strong>9.2</strong></td>
<td><strong>14</strong></td>
<td><strong>27</strong></td>
<td><strong>37</strong></td>
<td><strong>35</strong></td>
<td><strong>31</strong></td>
<td><strong>52</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SafePBNF**

<table>
<thead>
<tr>
<th></th>
<th>threads</th>
<th>logistics-6</th>
<th>blocks-14</th>
<th>gripper-7</th>
<th>satellite-6</th>
<th>elevator-12</th>
<th>freecell-3</th>
<th>depots-7</th>
<th>driverlog-11</th>
<th>gripper-8</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.1</td>
<td>7.8</td>
<td>42</td>
<td>62</td>
<td>152</td>
<td>131</td>
<td>167</td>
<td>152</td>
<td>243</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.1</td>
<td>4.3</td>
<td>18</td>
<td>24</td>
<td>59</td>
<td>57</td>
<td>67</td>
<td>62</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.79</td>
<td>3.9</td>
<td>12</td>
<td>20</td>
<td>41</td>
<td>48</td>
<td>48</td>
<td>43</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.71</td>
<td>3.4</td>
<td>10</td>
<td><strong>14</strong></td>
<td>32</td>
<td>45</td>
<td>43</td>
<td>35</td>
<td>59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BFPSS**

<table>
<thead>
<tr>
<th></th>
<th>threads</th>
<th>logistics-6</th>
<th>blocks-14</th>
<th>gripper-7</th>
<th>satellite-6</th>
<th>elevator-12</th>
<th>freecell-3</th>
<th>depots-7</th>
<th>driverlog-11</th>
<th>gripper-8</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.1</td>
<td>7.8</td>
<td>42</td>
<td>62</td>
<td>152</td>
<td>131</td>
<td>167</td>
<td>152</td>
<td>243</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.1</td>
<td>4.3</td>
<td>18</td>
<td>24</td>
<td>59</td>
<td>57</td>
<td>67</td>
<td>62</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.79</td>
<td>3.9</td>
<td>12</td>
<td>20</td>
<td>41</td>
<td>48</td>
<td>48</td>
<td>43</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.71</td>
<td>3.4</td>
<td>10</td>
<td><strong>14</strong></td>
<td>32</td>
<td>45</td>
<td>43</td>
<td>35</td>
<td>59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- PBNF gives the best performance and scalability across all domains tested.

- Overhead of livelock prevention procedure seems low.

- Simple to apply to:
  - non-unit cost domains.
  - any $f(n)$ ordering.
Bounded Suboptimal Search
Weighted PBNF

- Sort open lists on $f'(n) = g(n) + w \cdot h(n)$.
- Terminate when open list is empty or minimum $f'(n) \geq g(s)$ for incumbent solution $s$.
- Solution returned guaranteed to be within $w$ factor of optimal.
- \textit{wA*} can drop duplicates while maintaining bounded suboptimality. We can’t.
wA* can drop duplicates while maintaining bounded suboptimality. We can’t.

Rule 1: Only re-expand duplicates if the new path is a factor \( w \) better.
Pruning When Search Order is not Strictly Best-First

- \( \text{wA}^* \) can drop duplicates while maintaining bounded suboptimality. We can’t.

- Rule 1: Only re-expand duplicates if the new path is a factor \( w \) better.

- Rule 2: Only retain nodes a factor of \( w \) better than incumbent. (See Burns et al in ICAPS ’09 for details.)
## Suboptimal Search

<table>
<thead>
<tr>
<th>w</th>
<th>wPBNF</th>
<th>wBFPSDD</th>
<th>wAPRA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>0.65</td>
<td>0.64</td>
<td>0.63</td>
</tr>
<tr>
<td>1.7</td>
<td>0.41</td>
<td>0.60</td>
<td>0.61</td>
</tr>
<tr>
<td>2.0</td>
<td>0.37</td>
<td>0.43</td>
<td>0.61</td>
</tr>
<tr>
<td>3.0</td>
<td>0.74</td>
<td>0.34</td>
<td>0.57</td>
</tr>
</tbody>
</table>
Sliding Tiles: wPBNF v.s wA*

log10(times faster than wA*)

log10(wA* expansions)

wpbnf-1.4 ○
wpbnf-1.8 +
wpbnf-2.0 w
wpbnf-3.0 <
wpbnf-5.0 s
### STRIPS Planning (Bounded Suboptimal)

#### Introduction

- PBNF
- Suboptimal Search
  - wPBNF
  - Pruning
  - Empirical Evaluation
- Summary

#### Suboptimal Search

<table>
<thead>
<tr>
<th>Problem</th>
<th>1.5</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>1.5</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>1.5</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>logistics-8</td>
<td>2.68</td>
<td>2.27</td>
<td>4.06</td>
<td>1.00</td>
<td>1.86</td>
<td>2.12</td>
<td>1.14</td>
<td>0.86</td>
<td>1.01</td>
<td>0.98</td>
<td>0.64</td>
<td>1.14</td>
</tr>
<tr>
<td>blocks-16</td>
<td>0.93</td>
<td>0.54</td>
<td>0.48</td>
<td>1.32</td>
<td>0.34</td>
<td>0.19</td>
<td>0.16</td>
<td><strong>2.42</strong></td>
<td>1.16</td>
<td>0.82</td>
<td><strong>2.42</strong></td>
<td>0.25</td>
</tr>
<tr>
<td>gripper-7</td>
<td>2.01</td>
<td>1.99</td>
<td>1.99</td>
<td><strong>2.02</strong></td>
<td>1.91</td>
<td>1.89</td>
<td>1.86</td>
<td>1.84</td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.75</td>
</tr>
<tr>
<td>satellite-6</td>
<td>2.02</td>
<td>1.53</td>
<td>5.90</td>
<td>3.04</td>
<td>1.71</td>
<td>2.22</td>
<td><strong>7.50</strong></td>
<td>2.80</td>
<td>0.71</td>
<td>0.84</td>
<td>0.67</td>
<td>0.79</td>
</tr>
<tr>
<td>elevator-12</td>
<td>2.02</td>
<td>2.08</td>
<td><strong>2.21</strong></td>
<td>2.15</td>
<td>1.76</td>
<td>1.76</td>
<td>1.81</td>
<td>2.18</td>
<td>0.70</td>
<td>0.68</td>
<td>0.72</td>
<td>0.73</td>
</tr>
<tr>
<td>freecell-3</td>
<td>2.06</td>
<td>0.84</td>
<td>8.11</td>
<td>10.7</td>
<td>1.42</td>
<td>0.54</td>
<td><strong>16.9</strong></td>
<td><strong>55.8</strong></td>
<td>1.12</td>
<td>1.18</td>
<td>0.84</td>
<td>1.03</td>
</tr>
<tr>
<td>depots-13</td>
<td>2.70</td>
<td><strong>4.49</strong></td>
<td>0.82</td>
<td>0.81</td>
<td>1.48</td>
<td>1.58</td>
<td>0.18</td>
<td>0.16</td>
<td>0.74</td>
<td>1.12</td>
<td>0.32</td>
<td>0.29</td>
</tr>
<tr>
<td>driverlog-11</td>
<td>0.85</td>
<td>0.19</td>
<td>0.69</td>
<td>0.62</td>
<td>0.85</td>
<td>0.11</td>
<td>0.19</td>
<td>0.21</td>
<td><strong>0.86</strong></td>
<td>0.33</td>
<td>0.20</td>
<td>0.19</td>
</tr>
<tr>
<td>gripper-8</td>
<td>2.06</td>
<td>2.04</td>
<td><strong>2.08</strong></td>
<td>2.07</td>
<td>2.00</td>
<td>1.96</td>
<td>1.97</td>
<td>1.98</td>
<td>0.61</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>logistics-8</td>
<td><strong>7.10</strong></td>
<td>6.88</td>
<td>1.91</td>
<td>0.46</td>
<td>3.17</td>
<td>3.59</td>
<td>0.11</td>
<td>0.1</td>
<td>3.17</td>
<td>2.77</td>
<td>2.23</td>
<td>1.02</td>
</tr>
<tr>
<td>blocks-16</td>
<td><strong>2.87</strong></td>
<td>0.7</td>
<td>0.37</td>
<td>1.26</td>
<td>0.49</td>
<td>0.22</td>
<td>0.01</td>
<td>0.32</td>
<td>2.67</td>
<td>1.06</td>
<td>1.02</td>
<td>0.13</td>
</tr>
<tr>
<td>gripper-7</td>
<td><strong>5.67</strong></td>
<td>5.09</td>
<td>5.07</td>
<td>5.18</td>
<td>4.33</td>
<td>4.28</td>
<td>4.14</td>
<td>4.05</td>
<td>1.76</td>
<td>1.75</td>
<td>1.74</td>
<td>1.80</td>
</tr>
<tr>
<td>satellite-6</td>
<td>4.42</td>
<td>2.85</td>
<td>2.68</td>
<td><strong>5.89</strong></td>
<td>3.13</td>
<td>2.31</td>
<td>3.01</td>
<td>1.05</td>
<td>1.10</td>
<td>0.96</td>
<td>1.18</td>
<td>0.97</td>
</tr>
<tr>
<td>elevator-12</td>
<td><strong>6.32</strong></td>
<td>6.31</td>
<td>7.10</td>
<td>3.68</td>
<td>3.78</td>
<td>4.04</td>
<td>3.95</td>
<td>0.94</td>
<td>0.96</td>
<td>1.03</td>
<td>1.01</td>
<td>1.01</td>
</tr>
<tr>
<td>freecell-3</td>
<td>7.01</td>
<td>2.31</td>
<td>131</td>
<td><strong>1721</strong></td>
<td>2.12</td>
<td>0.70</td>
<td>44.5</td>
<td>137</td>
<td>2.48</td>
<td>0.76</td>
<td>2.93</td>
<td>2.96</td>
</tr>
<tr>
<td>depots-13</td>
<td><strong>3.12</strong></td>
<td>1.80</td>
<td>0.87</td>
<td>0.88</td>
<td>1.88</td>
<td>1.87</td>
<td>0.13</td>
<td>0.12</td>
<td>M</td>
<td>M</td>
<td>2.58</td>
<td>0.12</td>
</tr>
<tr>
<td>driverlog-11</td>
<td><strong>1.72</strong></td>
<td>0.43</td>
<td>0.67</td>
<td>0.42</td>
<td>1.26</td>
<td>0.21</td>
<td>0.30</td>
<td>0.23</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>gripper-8</td>
<td><strong>5.85</strong></td>
<td>5.31</td>
<td>5.40</td>
<td>5.44</td>
<td>4.62</td>
<td>4.55</td>
<td>4.55</td>
<td>4.51</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

Seth Lemons (UNH) Parallel Best-First Search: Optimal and Suboptimal Solutions – 22 / 39
■ Higher weights reduce benefit of parallelizing.

■ Greater benefit on problems that require more work for wA*.
Anytime Search
■ Sort open lists on $f'(n) = g(n) + w \cdot h(n)$.

■ Continue after first goal, returning stream of improving solutions. (Hansen and Zhou, 2007)

■ Terminate when open list is empty or minimum $f(n) \geq g(s)$ for incumbent solution $s$.

■ Converges to optimal solution, proving optimality.
Four-way Grid Pathfinding (Anytime Raw)

Unit Four-way Grids: AwPBNF Raw Profiles

- Solution Cost (factor over optimal)
  - 1.04
  - 1.03
  - 1.02
  - 1.01
  - 1.0

- Wall time relative to serial A*
  - 0.8
  - 0.6
  - 0.4
  - 0.2

1.8
1.4
1.1
1.2
Four-way Grid Pathfinding (Anytime)

Introduction

PBNF

Suboptimal Search

Anytime Search

- AwPBNF
- Empirical Evaluation

Summary

Conclusion

Unit Four-way Grids

Solution Cost (factor over optimal)

Wall time relative to serial A*

AwA*
AwPBNF 2 threads
ARA*
AwPBNF 4 threads
AwPBNF 8 threads
<table>
<thead>
<tr>
<th></th>
<th>AwPBNF</th>
<th>AwBFPSDD</th>
<th>AwAPRA*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>logistics-6</td>
<td>1.06</td>
<td>1.35</td>
<td>1.94</td>
</tr>
<tr>
<td>blocks-14</td>
<td>1.91</td>
<td>1.99</td>
<td>13.22</td>
</tr>
<tr>
<td>gripper-7</td>
<td><strong>2.05</strong></td>
<td>1.96</td>
<td>1.99</td>
</tr>
<tr>
<td>satellite-6</td>
<td>1.58</td>
<td>1.96</td>
<td><strong>1.98</strong></td>
</tr>
<tr>
<td>elevator-12</td>
<td>2.01</td>
<td>2.07</td>
<td><strong>2.13</strong></td>
</tr>
<tr>
<td>freecell-3</td>
<td>1.93</td>
<td>1.06</td>
<td>2.78</td>
</tr>
<tr>
<td>depots-7</td>
<td>1.94</td>
<td>2.00</td>
<td>2.01</td>
</tr>
<tr>
<td>driverlog-11</td>
<td>1.95</td>
<td><strong>2.10</strong></td>
<td>1.99</td>
</tr>
<tr>
<td>gripper-8</td>
<td>2.04</td>
<td>2.05</td>
<td><strong>2.09</strong></td>
</tr>
<tr>
<td>logistics-6</td>
<td>2.04</td>
<td>2.46</td>
<td>4.19</td>
</tr>
<tr>
<td>blocks-14</td>
<td>3.72</td>
<td>22.4</td>
<td>25.7</td>
</tr>
<tr>
<td>gripper-7</td>
<td><strong>5.61</strong></td>
<td>5.05</td>
<td>5.03</td>
</tr>
<tr>
<td>satellite-6</td>
<td><strong>5.96</strong></td>
<td>4.66</td>
<td>5.74</td>
</tr>
<tr>
<td>elevator-12</td>
<td>6.18</td>
<td>6.03</td>
<td><strong>6.20</strong></td>
</tr>
<tr>
<td>freecell-3</td>
<td>3.54</td>
<td>1.50</td>
<td><strong>15.3</strong></td>
</tr>
<tr>
<td>depots-7</td>
<td>5.74</td>
<td>5.52</td>
<td>5.48</td>
</tr>
<tr>
<td>driverlog-11</td>
<td>5.78</td>
<td><strong>5.83</strong></td>
<td>5.73</td>
</tr>
<tr>
<td>gripper-8</td>
<td><strong>5.82</strong></td>
<td>5.36</td>
<td>5.39</td>
</tr>
</tbody>
</table>
Best-case performance better than serial in all domains tested.

Higher weights sometimes hurt performance.
Conclusion
Parallel Best $N$Block First:

- Fast
  - Beats all other algorithms used for comparison when returning optimal solutions.
  - More speedup for larger problems in suboptimal or anytime setting.

- Scales well
  - Tested out to eight threads.

- Easy to use
  - Only requires a user-provided abstraction.

- New pruning rules for suboptimal search.

For more information on optimal results, see IJCAI session on “Advances in A* Search” on Thursday, July 16 at 4PM.
Tell your students to apply to grad school in CS at UNH!

- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, systems
Additional Slides
### Four-way Grid Pathfinding (Bounded Suboptimal)

<table>
<thead>
<tr>
<th>w</th>
<th>wPBNF</th>
<th>wBFPSDD</th>
<th>wAPRA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>0.83</td>
<td>1.31</td>
<td>0.93</td>
</tr>
<tr>
<td>1.2</td>
<td>0.76</td>
<td>1.24</td>
<td>0.88</td>
</tr>
<tr>
<td>1.4</td>
<td>0.52</td>
<td>1.15</td>
<td>0.55</td>
</tr>
<tr>
<td>1.8</td>
<td>0.53</td>
<td>0.73</td>
<td>0.51</td>
</tr>
</tbody>
</table>
Easy Sliding 15-Puzzles (Anytime)

Introduction

PBNF

Suboptimal Search

Anytime Search

Conclusion

Additional Slides

Easy Korf 15-puzzles

Wall time relative to serial A*

Solution Cost (factor over optimal)

AwPBNF 4 threads
AwPBNF 8 threads
AwPBNF 2 threads
AwA*
ARA*
### STRIPS Planning (Anytime vs PBNF)

#### Introduction

#### PBNF

#### Suboptimal Search

#### Anytime Search

#### Conclusion

#### Additional Slides

<table>
<thead>
<tr>
<th>Threads</th>
<th>AwPBNF</th>
<th>AwBFPSDD</th>
<th>AwAPRA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>2.36</td>
<td>1.46</td>
<td>1.08</td>
</tr>
<tr>
<td>2</td>
<td>2.27</td>
<td>1.42</td>
<td>1.08</td>
</tr>
<tr>
<td>3</td>
<td>0.71</td>
<td>0.94</td>
<td>1.08</td>
</tr>
<tr>
<td>5</td>
<td>0.54</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threads</th>
<th>AwPBNF</th>
<th>AwBFPSDD</th>
<th>AwAPRA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>1.08</td>
<td>1.42</td>
<td>1.08</td>
</tr>
<tr>
<td>2</td>
<td>1.08</td>
<td>1.08</td>
<td>1.08</td>
</tr>
<tr>
<td>3</td>
<td>0.98</td>
<td>0.94</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Seth Lemons (UNH)  Parallel Best-First Search: Optimal and Suboptimal Solutions – 36 / 39
Eight-way Grid Pathfinding (Optimal)

Introduction

PBNF

Suboptimal Search

Anytime Search

Conclusion

Additional Slides

Grid Unit 8-Way

wall time (seconds)

threads

Serial A* ---
BFPSDD ------
APRA* -------
SafePBNF ----
PBNF ----

Seth Lemons (UNH) Parallel Best-First Search: Optimal and Suboptimal Solutions – 37 / 39
<table>
<thead>
<tr>
<th>w</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>0.57</td>
<td>1.08</td>
<td>1.70</td>
<td>1.79</td>
<td>1.75</td>
<td>0.69</td>
<td>1.06</td>
<td>1.47</td>
<td>1.53</td>
<td>1.62</td>
<td>0.67</td>
<td>1.05</td>
<td>1.67</td>
<td>1.87</td>
<td>1.58</td>
</tr>
<tr>
<td>1.2</td>
<td>0.29</td>
<td>0.28</td>
<td>0.26</td>
<td>0.25</td>
<td>0.24</td>
<td>0.38</td>
<td>0.42</td>
<td>0.39</td>
<td>0.36</td>
<td>0.29</td>
<td>0.78</td>
<td>0.64</td>
<td>0.43</td>
<td>0.36</td>
<td>0.26</td>
</tr>
<tr>
<td>1.4</td>
<td>0.15</td>
<td>0.14</td>
<td>0.13</td>
<td>0.13</td>
<td>0.12</td>
<td>0.26</td>
<td>0.34</td>
<td>0.30</td>
<td>0.27</td>
<td>0.22</td>
<td>0.77</td>
<td>0.51</td>
<td>0.33</td>
<td>0.28</td>
<td>0.19</td>
</tr>
<tr>
<td>1.8</td>
<td>0.14</td>
<td>0.13</td>
<td>0.12</td>
<td>0.12</td>
<td>0.11</td>
<td>0.29</td>
<td>0.28</td>
<td>0.24</td>
<td>0.22</td>
<td>0.18</td>
<td>0.77</td>
<td>0.51</td>
<td>0.33</td>
<td>0.28</td>
<td>0.20</td>
</tr>
</tbody>
</table>
Unit Eight-way Grids

![Graph showing solution cost and wall time relative to serial A* for different algorithms: ARA*, AwPBNF 2 threads, AwPBNF 4 threads, AwPBNF 8 threads. The graph illustrates the performance of these algorithms over wall time, with the y-axis representing solution cost (factor over optimal) and the x-axis representing wall time relative to serial A*.](attachment://graph.png)