Analyzing and Avoiding Pathological Behavior in Parallel Best-First Search

Ryo Kuroiwa (University of Toronto)
Alex Fukunaga (University of Tokyo)
Parallel Best-First Search (BFS)

- BFS is used to solve graph search problems such as planning
- Many parallelization methods have been previously proposed
- Previous parallel A* methods have been experimentally shown to scale well

<table>
<thead>
<tr>
<th>Method</th>
<th>BFS</th>
<th>Environment</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDA*</td>
<td>A*</td>
<td>distributed</td>
<td>Kishimoto et al. 2009</td>
</tr>
<tr>
<td>PBNF</td>
<td>A*, WA*</td>
<td>multi-core</td>
<td>Burns et al. 2010</td>
</tr>
<tr>
<td>KPBFS</td>
<td>WA*</td>
<td>multi-core</td>
<td>Vidal et al. 2010</td>
</tr>
<tr>
<td>HDGBFS, LE, LG</td>
<td>GBFS</td>
<td>distributed</td>
<td>Kuroiwa and Fukunaga 2019</td>
</tr>
</tbody>
</table>
Parallel Best-First Search (BFS)

- BFS is used to solve graph search problems such as planning
- Many parallelization methods have been previously proposed
- Previous parallel A* methods have been experimentally shown to scale well

- What about parallel Greedy Best-First Search (GBFS)?

<table>
<thead>
<tr>
<th>Method</th>
<th>BFS</th>
<th>Environment</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDA*</td>
<td>A*</td>
<td>distributed</td>
<td>Kishimoto et al. 2009</td>
</tr>
<tr>
<td>PBNF</td>
<td>A*, WA*</td>
<td>multi-core</td>
<td>Burns et al. 2010</td>
</tr>
<tr>
<td>KPBFS</td>
<td>WA*</td>
<td>multi-core</td>
<td>Vidal et al. 2010</td>
</tr>
<tr>
<td>HDGBFS, LE, LG</td>
<td>GBFS</td>
<td>distributed</td>
<td>Kuroiwa and Fukunaga 2019</td>
</tr>
</tbody>
</table>
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from \( s_0 \) to \( s^* \)
- GBFS expands \( s \) with min. \( h(s) \) (a heuristic value)

\[
\begin{align*}
\text{GBFS expands } s & \text{ with min. } h(s) \text{ (a heuristic value)} \\
\end{align*}
\]
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)

```
h(s_1) = 2  h(s_3) = 3  h(s_5) = 1  h(s^*) = 0
```

```
h(s_2) = 1  h(s_4) = 3  h(s_6) = 2
```

```
open
```

```
s_0
```

```
h(s_0) = 3
```

```
s_1
```

```
h(s_1) = 2
```

```
s_3
```

```
h(s_3) = 3
```

```
s_5
```

```
h(s_5) = 1
```

```
s^*
```

```
h(s^*) = 0
```

```
s_6
```

```
h(s_6) = 2
```

```
s_4
```

```
h(s_4) = 3
```
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from \( s_0 \) to \( s^* \)
- GBFS expands \( s \) with \( \min h(s) \) (a heuristic value)

![Graph Search Problem Example Diagram]

\[
\begin{align*}
  h(s_0) &= 3 \\
  h(s_1) &= 2 \\
  h(s_2) &= 1 \\
  h(s_3) &= 3 \\
  h(s_4) &= 3 \\
  h(s_5) &= 1 \\
  h(s_6) &= 2 \\
  h(s^*) &= 0
\end{align*}
\]
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)

The tie-breaking strategy decides which to expand.
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)
A Graph Search Problem Example with Greedy Best-First Search (GBFS)

- A solution of a graph search problem is a path from $s_0$ to $s^*$
- GBFS expands $s$ with min. $h(s)$ (a heuristic value)
A solution of a graph search problem is a path from $s_0$ to $s^*$

- GBFS expands $s$ with min. $h(s)$ (a heuristic value)
Pathological Behavior in Parallel GBFS
[Kuroiwa and Fukunaga 2019]

- GBFS, HDGBFS, LE, and LG were experimentally compared with the 5 min. time limit for each instance
Pathological Behavior in Parallel GBFS
[Kuroiwa and Fukunaga 2019]

● GBFS, HDGBFS, LE, and LG were experimentally compared with the 5 min. time limit for each instance

● Parallel GBFS methods **failed to solve easy instances** which GBFS solved within 1 sec. and 100 expansions
Pathological Behavior in Parallel GBFS
[Kuroiwa and Fukunaga 2019]

- GBFS, HDGBFS, LE, and LG were experimentally compared with the 5 min. time limit for each instance
- Parallel GBFS methods **failed to solve easy instances** which GBFS solved within 1 sec. and 100 expansions
- Parallel GBFS methods **expanded >1000 times as many states** as GBFS in some instances
Pathological Behavior in Parallel GBFS
[Kuroiwa and Fukunaga 2019]

- GBFS, HDGBFS, LE, and LG were experimentally compared with the 5 min. time limit for each instance
- Parallel GBFS methods **failed to solve easy instances** which GBFS solved within 1 sec. and 100 expansions
- Parallel GBFS methods **expanded >1000 times as many states** as GBFS in some instances
- **Can we obtain theoretical bound for the performance degradation?**
KBFS: a Model of Parallel BFS
[Kuroiwa and Fukunaga 2019]

- KBFS [Felner et al. 2003]: similar to BFS, but simultaneously expands $k$ states
- HDA*, KPBFS, HDGBFS, and LE can be modeled as KBFS
KBFS: a Model of Parallel BFS

[Kuroiwa and Fukunaga 2019]

- KBFS [Felner et al. 2003]: similar to BFS, but simultaneously expands \( k \) states
- HDA*, KPBFS, HDGBFS, and LE can be modeled as KBFS

\[ \begin{align*}
  k = 2 & & h(s_1) = 2 & & h(s_3) = 3 & & h(s_5) = 1 & & h(s^*) = 0 \\
  h(s_0) = 3 & & h(s_2) = 1 & & h(s_4) = 3 & & h(s_6) = 2
\end{align*} \]
KBFS: a Model of Parallel BFS
[Kuroiwa and Fukunaga 2019]

- KBFS [Felner et al. 2003]: similar to BFS, but simultaneously expands $k$ states
- HDA*, KPBFS, HDGBFS, and LE can be modeled as KBFS

```
k = 2

h(s_0) = 3
h(s_1) = 2
h(s_2) = 1
h(s_3) = 3
h(s_4) = 3
h(s_5) = 1
h(s_6) = 2
h(s^*) = 0
```
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

**GBFS expands 6 states**

- $h(s_1) = 1$
- $h(s_3) = 2$
- $h(s_5) = 2$
- $h(s_7) = 1$
- $h(s^*) = 0$

Diagram:

- $h(s_0) = 2$
- $h(s_2) = 3$
- $h(s_4) = 2$
- $h(s_8) = 2$
- $i = 1, \ldots, n$
- $h(s_6^i) = 1$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with $1 < k \leq n$

$h(s_1) = 1$  $h(s_3) = 2$  $h(s_5) = 2$  $h(s_7) = 1$  $h(s^*) = 0$

$h(s_0) = 2$  $h(s_2) = 3$  $h(s_4) = 2$  $h(s_6^i) = 1$  $h(s_8) = 2$

$i = 1, \ldots, n$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with $1 < k \leq n$

$h(s_0) = 2$

$h(s_1) = 1$
$h(s_3) = 2$
$h(s_5) = 2$
$h(s_7) = 1$
$h(s^*) = 0$

$h(s_2) = 3$
$h(s_4) = 2$
$h(s_6^i) = 1$
$h(s_8) = 2$

$s_0 \rightarrow s_1$ and $s_2$

$s_1 \rightarrow s_3$
$s_2 \rightarrow s_4$

$s_3 \rightarrow s_5$
$s_4 \rightarrow s_6^i$

$s_5 \rightarrow s_7$
$s_6^i \rightarrow s_8$

$s_7 \rightarrow s^*$

$i = 1, \ldots, n$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with $1 < k \leq n$

- $h(s_1) = 1$
- $h(s_3) = 2$
- $h(s_5) = 2$
- $h(s_7) = 1$
- $h(s^{*}) = 0$

- $h(s_0) = 2$
- $h(s_2) = 3$
- $h(s_4) = 2$
- $h(s_8) = 2$

For $i = 1, \ldots, n$:
- $h(s_6^i) = 1$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with $1 < k \leq n$

$$
\begin{align*}
    h(s_1) &= 1 \\
    h(s_3) &= 2 \\
    h(s_5) &= 2 \\
    h(s_7) &= 1 \\
    h(s^*) &= 0 \\
    h(s_0) &= 2 \\
    h(s_2) &= 3 \\
    h(s_4) &= 2 \\
    h(s_6^i) &= 1 \\
    i &= 1, \ldots, n
\end{align*}
$$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with $1 < k \leq n$ expands more than $n$ states: $n \to \infty$

$h(s_1) = 1 \quad h(s_3) = 2 \quad h(s_5) = 2 \quad h(s_7) = 1 \quad h(s^*) = 0$

$h(s_0) = 2 \quad h(s_2) = 3 \quad h(s_4) = 2 \quad h(s_8) = 2$

$i = 1, \ldots, n \quad h(s_6^i) = 1$
KGBFS (a GBFS Version of KBFS) can expand arbitrarily more states than GBFS

KGBFS with \( 1 < k \leq n \) expands more than \( n \) states: \( n \to \infty \)

Regardless of the tie-breaking strategy

\[
\begin{align*}
&h(s_1) = 1 & h(s_3) = 2 & h(s_5) = 2 & h(s_7) = 1 & h(s^*) = 0 \\
&h(s_0) = 2 & h(s_2) = 3 & h(s_4) = 2 & h(s_8) = 2 \\
&i = 1, \ldots, n & h(s_6^i) = 1
\end{align*}
\]
Pathology and $t$-Boundedness

- $A$ is **pathological** relative to $B$ if $A$ expands arbitrarily more states than $B$ does given some graph and heuristic.

- $A$ is **$t$-bounded** relative to $B$ if $A$ expands no more than $t$ times as many states as $B$ does, for any graph and heuristic.
## Pathologies in Parallel BFS

<table>
<thead>
<tr>
<th>Method</th>
<th>Heuristic</th>
<th>$t$-boundedness</th>
<th>Model</th>
<th>BFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDA* ($k$ processes)</td>
<td>consistent</td>
<td>$k$-bounded</td>
<td>KBFS (KA*)</td>
<td>A*</td>
</tr>
<tr>
<td>HDA*</td>
<td>inconsistent</td>
<td>pathological</td>
<td>KBFS (KA*)</td>
<td>A*</td>
</tr>
<tr>
<td>KPBFS ($w &gt; 1$)</td>
<td></td>
<td>pathological</td>
<td>KBFS (KWA*)</td>
<td>WA*</td>
</tr>
<tr>
<td>KPBFS, HDGBFS, LE</td>
<td></td>
<td>pathological</td>
<td>KGBFS</td>
<td>GBFS</td>
</tr>
<tr>
<td>LG</td>
<td></td>
<td>pathological</td>
<td>KGBFS-like</td>
<td>GBFS</td>
</tr>
</tbody>
</table>
KA* is \( k \)-Bounded Relative to A* with a consistent heuristic

- A heuristic is **consistent**: \( h(s') \geq h(s) + c(s, s') \)
  \( c(s, s') \) is the cost of the edge \((s, s')\)
- KA* with any tie-breaking strategy is \( k \)-bounded relative to A* with the worst-case tie-breaking strategy

**Proof Sketch:** A* expands \( s \) with \( f(s) = g(s) + h(s) \) where \( g(s) \) is the cost of the path from \( s_0 \) to \( s \). A* expands each \( s \) with \( f(s) < f^* \) (the cost of the optimal path). KA* expands \( s \) with \( f(s) \leq f^* \) at every \( k \) expansions because \( s' \) on an optimal path has \( f(s') \leq f^* \) by the consistency.
**TB-Boundedness:** Another Type of Bound

A is **TB-bounded** relative to B if A expands only states expanded by B with some tie-breaking strategy.

<table>
<thead>
<tr>
<th>Method</th>
<th>TB-boundedness</th>
<th>BFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDA*</td>
<td>no</td>
<td>A*</td>
</tr>
<tr>
<td>KPBFS ($w &gt; 1$)</td>
<td>no</td>
<td>WA*</td>
</tr>
<tr>
<td>KPBFS, HDGBFS, LE, LG</td>
<td>no</td>
<td>GBFS</td>
</tr>
</tbody>
</table>
$P_{\text{GBFS}}/C$: a $k$-Bounded Parallel GBFS

- $P_{\text{GBFS}}$ executes $k$ threads of independent GBFS with different tie-breaking strategies in parallel [Kuroiwa and Fukunaga 2019] Each GBFS does not change its expansion order
- $P_{\text{GBFS}}/C$ uses the shared evaluation cache of heuristic values to speed up each GBFS while keeping the expansion order
- $P_{\text{GBFS}}$ and $P_{\text{GBFS}}/C$ are also $TB$-bounded
SPUHF: a $TB$-Bounded Parallel GBFS

- PUHF (Parallel Under High-water mark First) expands $s$ only if $h(s) \leq h(parent(s))$ or any other thread is not expanding a state.

- Proof Sketch: If $h(s) \leq h(parent(s))$, $s$ is expanded by GBFS with some tie-breaking strategy.

- SPUHF (Speculative PUHF) executes independent parallel search (speculative search) using idling threads with the shared evaluation cache.
Bounded Parallel GBFS

- $P_{\text{GBFS}}/C$: $k$-bounded and $TB$-bounded
  A parallel portfolio of independent GBFS with different tie-breaking strategies using the shared evaluation cache $k$-bounded and $TB$-bounded

- SPUHF: $TB$-bounded
  A multi-core parallel GBFS similar to KPBFS

- See the paper for details
### Experimental Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Coverage</th>
<th># of solved instances unsolved by GBFS</th>
<th># of unsolved instances solved by GBFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG</td>
<td>888</td>
<td>137</td>
<td>13</td>
</tr>
<tr>
<td>KPBFS</td>
<td>880</td>
<td>135</td>
<td>19</td>
</tr>
<tr>
<td>$P_{\text{GBFS/C}}$</td>
<td>928</td>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>SPUHF</td>
<td>864</td>
<td>115</td>
<td>15</td>
</tr>
</tbody>
</table>

Domain-wise results are complementary

55 domains from IPC-98-18
5 min. time limit
122 GiB memory limit
### Conclusion/Summary of Contributions

- Proposed $t$-boundedness, Pathology, $TB$-boundedness
- Analyzed existing parallel BFS
- Proposed $P_{GBFS}/C$ and SPUHF

<table>
<thead>
<tr>
<th>Method</th>
<th>$t$-boundedness</th>
<th>$TB$-boundedness</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDA* [Kishimoto et al. 2009] ($k$ processes), consistent heuristic</td>
<td>$k$-bounded</td>
<td>no</td>
</tr>
<tr>
<td>HDA*, inconsistent heuristic</td>
<td>pathological</td>
<td>no</td>
</tr>
<tr>
<td>KPBFS ($w &gt; 1$) [Vidal et al. 2010]</td>
<td>pathological</td>
<td>no</td>
</tr>
<tr>
<td>KPBFS, HDGBFS, LE, LG [Kuroiwa and Fukunaga 2019]</td>
<td>pathological</td>
<td>no</td>
</tr>
<tr>
<td>$P_{GBFS}/C$ [New]</td>
<td>$k$-bounded</td>
<td>$TB$-bounded</td>
</tr>
<tr>
<td>SPUHF [New]</td>
<td>unknown</td>
<td>$TB$-bounded</td>
</tr>
</tbody>
</table>