Iterative Budgeted Exponential Search

Malte Helmert\textsuperscript{1}, Tor Lattimore\textsuperscript{2}, Levi H.S. Lelis\textsuperscript{3}, Laurent Orseau\textsuperscript{2}, Nathan Sturtevant\textsuperscript{4}

\textsuperscript{1}University of Basel, Switzerland
\textsuperscript{2}DeepMind, UK
\textsuperscript{3}Federal de Viçosa, Brazil
\textsuperscript{4}University of Alberta, Canada

IJCAI 2019 - Macau, China

Talk jointly presented by Laurent Orseau and Nathan Sturtevant
h = 11
Shortest Path Algorithm

History

1960

1970

1980

1990

2000

2010

Dijkstra’s Algorithm

A*

A* worst case†

IDA*

IDA* worst case†

IBEX
The $N^2$ problem

- Algorithms perform $N^2$ re-expansions of $N$ states
- IDA*: $N^2$ expansions of $N = b^d$ states
- A*: $2^N$ expansions of $N$ states
  - B & B’: $N^2$ (Martelli 1977; Mero, 1984)
- Weighted A*: $N^2$ (Chen et al, 2019)
IDA* Refresher

• IDA* does *iterative deepening* search on *f*-costs
  • \( f(n) = g(n) + h(n) \)

• Next iteration *f*-cost:
  • Smallest unexplored from previous iteration

• If *f*-cost layers grow exponentially
  • IDA* is asymptotically optimal in node expansions
IDA* - Unit Costs
2 States
x8
16 States
x5
79 States

f-cost 11

f-cost 13

f-cost 15
IDA* Worst Case

• $f$-cost layers grow exponentially
  • $1 + b + b^2 + b^3 + \ldots + b^d \approx b^d$

• What if $f$-cost layers grew linearly?
  • $1 + 2 + 3 + 4 + \ldots + b^d \approx (b^d)^2$

• Happens with non-unit edge costs:
  • STP: Cost of moving tile $t$: $\frac{t + 2}{t + 1}$
Previous Work

• Iterative Deepening
  • IDA* (Korf, 1985)
  • IDA*\textsubscript{CR} (Sarkar et al, 1990)
  • IDA*\textsubscript{IM} (Burns & Ruml, 2013)
  • EDA* (Sharon et al, 2014)
A* (B/B’)

• A* with a consistent heuristic is optimal*
• With an inconsistent heuristic A* might do $2^N$ expansions of N states
• B and B’ immediately propagate shorter paths
  • Will do at most $N^2$ expansions of N states
N states

N states
Previous Work

- Inconsistent Heuristics
  - B/B’ (Martelli 1977; Mero, 1984)
  - Delay (Sturtevant et al, 2008)
  - BPMX (Felner et al, 2011)
Demos

• All demos shown here runnable online:
  • https://www.movingai.com/SAS/BTS/

• Also running on my iPad; can give detailed demo
IDA* — Good Case

Number of expansions vs. f-cost max
IDA* — Good Case

![Graph showing the number of expansions vs. f-cost max. The graph has a horizontal axis labeled 'f-cost max' and a vertical axis labeled 'Number of expansions.' There are data points marked with 'N' and 'C*.' The graph suggests a good case scenario for IDA*.](image-url)
IDA* — Good Case

Number of expansions vs. f-cost max
IDA* — Good Case

The diagram shows a scatter plot with the x-axis labeled "f-cost max" and the y-axis labeled "Number of expansions". The plot includes markers at specific points, indicating the number of expansions at different f-cost max values. The markers at f-cost max 9 and 10 are highlighted, with the number of expansions marked as N and C* respectively.
IDA* — Bad Case

Number of expansions

f-cost max

1

33

36

$10^{10}$
IDA* — Bad Case

Number of expansions

\[ 10^{10} \]

\[ N = 36 \]

f-cost max

\[ 2 \]

\[ 33 \]

C*
IDA* — Bad Case
IDA* — Bad Case

Graph showing the number of expansions vs. f-cost max. The graph has a logarithmic scale on the y-axis and a linear scale on the x-axis. The x-axis is labeled "f-cost max" and the y-axis is labeled "Number of expansions." There are several data points marked on the graph, including a point at (33, 36) and a point at (4, N), indicating a steep increase in the number of expansions as f-cost max increases.
IDA* — Bad Case

The graph shows the relationship between the number of expansions and the f-cost max. The x-axis represents the f-cost max, ranging from 5 to 33, and the y-axis represents the number of expansions, ranging from 10^0 to 10^10. The data points indicate a linear relationship, with the number of expansions increasing as the f-cost max increases.
IDA* — Bad Case

Number of expansions

$10^{10}$

f-cost max

N

36

expansions

C*

30 33
IDA* — Bad Case

Number of expansions

36

N

expansions

3133

f-cost max

$10^{10}$
IDA* — Bad Case
Number of expansions

10^{10}

f-cost max

N

36

C*

33
For budget = 2, 4, 8, 16, ...:
    cost_bound = Oracle(budget)
    search within cost_bound and within budget

Oracle(budget) returns largest cost for which budget is sufficient.
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td></td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td></td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td>4</td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>13.9</td>
<td>8</td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>13.9</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>14.4</td>
<td>16</td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>13.9</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>14.4</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>16.8</td>
<td>30</td>
</tr>
</tbody>
</table>
IBEX (main idea): With an oracle

<table>
<thead>
<tr>
<th>budget</th>
<th>$f$-cost</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>13.5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>13.9</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>14.4</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>16.8</td>
<td>30</td>
</tr>
<tr>
<td>64</td>
<td>17.7</td>
<td>62</td>
</tr>
</tbody>
</table>
For budget = 2, 4, 8, 16, ...:
  cost_bound = Oracle(budget)
  search within cost_bound and within budget
IBEX (main idea)

For budget = 2, 4, 8, 16, ...:
  cost_bound = Oracle(budget)
  search within cost_bound and within budget

Oracle(budget) returns largest cost for which budget is sufficient.

Oracle(budget) = (budgeted) exponential search
IBEX (main idea)
IBEX (main idea)
IBEX (main idea)
IBEX (main idea)
IBEX (main idea)
IBEX (main idea)
IBEX (main idea)
For budget = 2, 4, 8, 16, 32, 64: \[ \leq 4N \]

With **exponential search**:  
Refine lower and upper bound on cost  
Proves no solution when lower = upper  

\[ O(\log C^*) \]
IBEX + Depth-First Search = *Budgeted Tree Search (BTS)*

<table>
<thead>
<tr>
<th>Alg.</th>
<th>Worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDA*_{(1985)}</td>
<td>$\Theta(N^2)$</td>
</tr>
<tr>
<td>EDA*_{(2014)}</td>
<td>unbounded</td>
</tr>
<tr>
<td>BTS</td>
<td>$O(N \log C^*)$</td>
</tr>
</tbody>
</table>

$C^*$: cost of the optimal solution  

$N$: number of nodes of cost at most $C^*$

All algorithms return an optimal solution (under usual assumptions).
IBEX + Uniform-Cost Search = *Budgeted Graph Search (BGS)*

<table>
<thead>
<tr>
<th>Alg.</th>
<th>Admissible Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Consistent</td>
</tr>
<tr>
<td>$A^*$ (1968)</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$B$ (1977)</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>BGS</td>
<td>$O(N)^\dagger$</td>
</tr>
</tbody>
</table>

$C^*$: cost of the optimal solution  
$N$: number of nodes of cost at most $C^*$

All algorithms return an optimal solution (under usual assumptions).
Enhancements: Tree search

- Recover IDA* when exponential unit-cost domains
  1) Infinite iteration at next $f$-cost
  2) If expansions grow exponentially, go to 1) else exponential search
Enhancements: Tree search

- Recover IDA* when exponential unit-cost domains
  1) Infinite iteration at next $f$-cost
  2) If expansions grow exponentially, go to 1) else exponential search
- Budget window $[2, 8]$ to be more aggressive
### Experiments

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>15-Puzzle (unit)</th>
<th>15-Puzzle (real)</th>
<th>Coconut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Exp. $\times 10^6$</td>
<td>Solved</td>
</tr>
<tr>
<td><strong>BTS</strong></td>
<td>100</td>
<td>242.5</td>
<td>100</td>
</tr>
<tr>
<td><strong>EDA</strong>*</td>
<td>99</td>
<td>5 586.0</td>
<td>100</td>
</tr>
<tr>
<td><em><em>IDA</em>^{CR}</em>*</td>
<td>100</td>
<td>868.4</td>
<td>100</td>
</tr>
<tr>
<td><strong>IDA</strong>*</td>
<td>100</td>
<td>242.5</td>
<td>57</td>
</tr>
<tr>
<td><strong>N</strong></td>
<td>100</td>
<td>100.8</td>
<td>100</td>
</tr>
</tbody>
</table>

**BTS** (*Budgeted Tree Search*) = IBEX + Depth-First Search
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>15-Puzzle (unit)</th>
<th>15-Puzzle (real)</th>
<th>Coconut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Exp. × 10^6</td>
<td>Solved</td>
</tr>
<tr>
<td>BTS</td>
<td>100</td>
<td>242.5</td>
<td>100</td>
</tr>
<tr>
<td>EDA*</td>
<td>99</td>
<td>5 586.0</td>
<td>100</td>
</tr>
<tr>
<td>IDA*_CR</td>
<td>100</td>
<td>868.4</td>
<td>100</td>
</tr>
<tr>
<td>IDA*</td>
<td>100</td>
<td>242.5</td>
<td>57</td>
</tr>
<tr>
<td>N</td>
<td>100</td>
<td>100.8</td>
<td>100</td>
</tr>
</tbody>
</table>

BTS (Budgeted Tree Search) = IBEX + Depth-First Search
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>15-Puzzle (unit)</th>
<th>15-Puzzle (real)</th>
<th>Coconut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Exp. × 10^6</td>
<td>Solved</td>
</tr>
<tr>
<td>BTS</td>
<td>100</td>
<td>242.5</td>
<td>100</td>
</tr>
<tr>
<td>EDA*</td>
<td>99</td>
<td>5586.0</td>
<td>100</td>
</tr>
<tr>
<td>IDA*_CR</td>
<td>100</td>
<td>868.4</td>
<td>100</td>
</tr>
<tr>
<td>IDA*</td>
<td>100</td>
<td>242.5</td>
<td>57</td>
</tr>
<tr>
<td>N</td>
<td>100</td>
<td>100.8</td>
<td>100</td>
</tr>
</tbody>
</table>

BTS (Budgeted Tree Search) = IBEX + Depth-First Search
IBEX: Budgeting → $O(N \log C)$

Tree search: drop-in replacement for IDA*

DovIBEX: For applications where oracle returns only if a solution is found
  - Interleaves budget iterations

Do you have a problem where the current worst case is $O(N^2)$, due to being cautious?

*Maybe (Dov)IBEX can help!*

Come and see our poster!