Faster than Weighted A*:
An Optimistic Approach to Bounded Suboptimal Search

Jordan Thayer and Wheeler Ruml

University of New Hampshire

{jtd7, ruml} at cs.unh.edu
Finding optimal solutions is prohibitively expensive.
Finding optimal solutions is prohibitively expensive.

Greedy solutions can be arbitrarily bad.

Grid Pathfinding

![Graph showing nodes generated vs problem size for A*, Greedy.

Four-way Grid Pathfinding (Unit cost)

![Graph showing solution cost (relative to A*) vs problem size for A*, Greedy.]
Finding optimal solutions is prohibitively expensive.

- Greedy solutions can be arbitrarily bad.
- Weighted A* bounds suboptimality.
Finding optimal solutions is prohibitively expensive.
• Greedy solutions can be arbitrarily bad.
• Weighted A* bounds suboptimality.
• Optimistic Search: faster search within the same bound.
Algorithm Overview

1: Greedy Phase
2: Cleanup Phase

Empirical Evaluation
Further Observations
Conclusion
Talk Outline

Algorithm Overview
Run weighted $A^*$ with a weight higher than the bound. Expand additional nodes to prove solution quality.

The Greedy Search Phase

The Cleanup Phase

Empirical Evaluation

Further Observations

Introduction

Algorithm Overview
- Predecessors
- Basic Idea

1: Greedy Phase

2: Cleanup Phase

Empirical Evaluation

Further Observations

Conclusion
Previous Algorithms: \( A^* \)

- A best first search expanding nodes in \( f \) order.
- \( f(n) = g(n) + h(n) \)
  If \( h(n) \) is admissible, returns optimal solution.
Previous Algorithms: Weighted $A^*$

- A best first search expanding nodes in $f'$ order.
- $f'(n) = g(n) + w \cdot h(n)$
  
  Solution quality bounded by $w$ for admissible $h(n)$.  

1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f$ value after a solution is found.

Continue until $w \cdot f_{min} > f(sol)$

This ’clean up’ guarantees solution quality.
1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f$ value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.
1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f$ value after a solution is found.
   Continue until $w \cdot f_{\text{min}} > f(\text{sol})$
   This 'clean up' guarantees solution quality.
1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f$ value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This ‘clean up’ guarantees solution quality.
1: Greedy Phase

- Weighted A*
Algorithm Overview

The Greedy Search Phase
Weighted $A^*$ becomes faster as the bound grows.
Weighted $A^*$ is often better than the bound.

The Cleanup Phase

Empirical Evaluation

Further Observations
$wA^*$ returns solutions faster as the bound increases.
Weighted $A^*$ is often better than the bound.

- $wA^*$ returns solutions better than the bound.
2: Cleanup Phase
Algorithm Overview

The Greedy Search Phase

The Cleanup Phase
Expand additional nodes in $f$ order.
Quit when the solution is provably within the bound.

Empirical Evaluation

Further Observations
Proving $w$-Admissibility

$w$-Admissibility

- $p$ is the deepest node on an optimal path to opt.
- $f_{\text{min}}$ is the node with the smallest $f$ value.
Proving $w$-Admissibility

- $p$ is the deepest node on an optimal path to opt.
- $f_{min}$ is the node with the smallest $f$ value.

$f_{min}$ provides a lower bound on solution cost.

Determine $f_{min}$ by priority queue sorted on $f$
Proving $w$-Admissibility

- $p$ is the deepest node on an optimal path to opt.
- $f_{min}$ is the node with the smallest $f$ value.

\[
\begin{align*}
    f(p) & \leq f(\text{opt}) \\
    f(f_{min}) & \leq f(p)
\end{align*}
\]

$f_{min}$ provides a lower bound on solution cost.

Determine $f_{min}$ by priority queue sorted on $f$.

Optimistic Search: Run a greedy search

Expand $f_{min}$ until $w \cdot f_{min} \geq f(sol)$
Proving \( w \)-Admissibility

- \( p \) is the deepest node on an optimal path to \( \text{opt} \).
- \( f_{\text{min}} \) is the node with the smallest \( f \) value.

\[
f(p) \leq f(\text{opt})
\]

\[
f(f_{\text{min}}) \leq f(p)
\]

\( f_{\text{min}} \) provides a lower bound on solution cost.

Determine \( f_{\text{min}} \) by priority queue sorted on \( f \).

Optimistic Search: Run a greedy search

Expand \( f_{\text{min}} \) until \( w \cdot f_{\text{min}} \geq f(\text{sol}) \)
Empirical Evaluation

Introduction

Algorithm Overview

1: Greedy Phase

2: Cleanup Phase

Empirical Evaluation

- Performance

Further Observations

Conclusion
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Algorithm Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Overview</td>
<td>The Greedy Search</td>
</tr>
<tr>
<td>1: Greedy Phase</td>
<td>Guaranteeing solution quality</td>
</tr>
<tr>
<td>2: Cleanup Phase</td>
<td>Empirical Evaluation</td>
</tr>
<tr>
<td>Empirical Evaluation</td>
<td>Results in several domains.</td>
</tr>
<tr>
<td>Further Observations</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Empirical Evaluation

- **Sliding Tile Puzzles**
  - Korf’s 100 15-puzzle instances (add date)

- **Traveling Salesman**
  - Unit Square
    - Pearl and Kim Hard (add date)

- **Grid world path finding**
  - Four-way and Eight-way Movement
  - Unit and Life Cost Models
    - 25%, 30%, 35%, 40%, 45% obstacles

- **Temporal Planning**
  - BlocksWorld, Logistics, Rover, Satellite, Zenotravel

See paper for additional plots.
Korf’s 15 Puzzles: $h = \text{Manhattan Distance}$

Node Generations Relative to IDA*

Sub-optimality bound
Performance of Optimistic Search

Introduction

Algorithm Overview

1: Greedy Phase

2: Cleanup Phase

Empirical Evaluation

- Performance

Further Observations

Conclusion

TSP: Pearl and Kim Hard

Node Generations Relative to A*

Sub-optimality bound

wA* - Optimistic

Jordan Thayer (UNH)
Performance of Optimistic Search

Introduction

Algorithm Overview

1: Greedy Phase

2: Cleanup Phase

Empirical Evaluation

■ Performance

Further Observations

Conclusion

Four-way Grid Pathfinding (Unit cost)

Nodes generated (relative to A*)

Sub-optimality Bound

wA*  
Optimistic
Performance of Optimistic Search

Introduction
Algorithm Overview
1: Greedy Phase
2: Cleanup Phase
Empirical Evaluation
Performance
Further Observations
Conclusion

logistics (problem 3)

Nodes generated (relative to A*)

Sub-optimality Bound

wA* ————
Optimistic Search ————
Further Observations

- Expansion Policy
- BAwA*

Conclusion
Talk Outline

- Algorithm Overview
- The Greedy Search
- Guaranteeing solution quality
- Empirical Evaluation
- Further Observations
  - Strict vs. Loose Expansion Policy
  - Bounded Anytime Weighted $A^*$
Strict Expansion Order:

- Algorithms like $wA^*$, $A^*_\epsilon$, Dynamically Weighted $A^*$
- Any expanded node can be shown to be within the bound at the time of their expansion
- Quality bound comes from this

Loose Expansion Order:

- Algorithms like Optimistic Search
- No restriction on the nodes expanded initially.
- Quality bound requires node expansion beyond the initial solution.
Anytime Heuristic Search:
Running weighted $A^*$ with a high weight
Continue node expansions after a solution is found
Anytime Heuristic Search:
Running weighted $A^*$ with a high weight
Continue node expansions after a solution is found

Bounded Anytime Weighted $A^*$:
Running weighted $A^*$ with a high weight
Continue node expansions after a solution is found
Add a second priority queue allows us to converge on a bound instead of on optimal.
1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f$ value after a solution is found. Continue until $w \cdot f_{\min} > f(sol)$
   This 'clean up' guarantees solution quality.
Bounded Anytime Weighted $A^*$ Expansions

1. Run weighted $A^*$ with a high weight.
2. Expand node with lowest $f'$ value after a solution is found.
   Continue until $w \cdot f_{min} > f(sol)$
   This 'clean up' guarantees solution quality.
Bounded Anytime Weighted A*

Korf’s 15 Puzzles

Node Generations Relative to IDA*

Sub-optimality bound

Optimistic

BAwA* — — — — —

wA* — — — — —

Conclusion

Empirical Evaluation

Further Observations

Expansion Policy

BAwA*
Bounded Anytime Weighted A*

Introduction

Algorithm Overview
1: Greedy Phase
2: Cleanup Phase

Empirical Evaluation

Further Observations
- Expansion Policy
- BAwA*

Conclusion

Pearl and Kim Hard

Node Generations Relative to A*

Sub-optimality bound

BAwA* - green
wA* - black
Optimistic - blue

Jordan Thayer (UNH)
Optimistic Search:

- Simple to implement.
- Performance is predictable.
- Current results are good, tuning could help.
  - Optimal greediness is still an open question.
- Consistently better than Weighted $A^*$
  - If you currently use $wA^*$, you should use Optimistic Search.
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, bioinformatics
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Algorithm Overview</td>
</tr>
<tr>
<td>1: Greedy Phase</td>
</tr>
<tr>
<td>2: Cleanup Phase</td>
</tr>
<tr>
<td>Empirical Evaluation</td>
</tr>
<tr>
<td>Further Observations</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
<tr>
<td>Additional Slides</td>
</tr>
<tr>
<td>■ Loose Bounds</td>
</tr>
<tr>
<td>■ Duplicates</td>
</tr>
<tr>
<td>■ Pseudo Code</td>
</tr>
</tbody>
</table>

**Additional Slides**
Weighted $A^*$ is often better than the bound.

- $wA^*$ returns solutions better than the bound.
Weighted $A^*$ Respects a Bound

\[ f(n) = g(n) + h(n) \]
\[ f'(n) = g(n) + w \cdot h(n) \]

\[
\begin{align*}
g(sol) & \leq f'(sol) \\
f'(sol) & \leq f'(p) \\
g(p) + w \cdot h(p) & \leq w \cdot (g(p) + h(p)) \\
w \cdot f(p) & \leq w \cdot f(opt) \\
w \cdot g(opt) & 
\end{align*}
\]

Therefore, $g(sol) \leq w \cdot g(opt)$
$f(n) = g(n) + h(n)$

$f'(n) = g(n) + w \cdot h(n)$

\[ g(sol) \]
\[ f'(sol) \leq f'(p) \]
\[ g(p) + w \cdot h(p) \leq w \cdot (g(p) + h(p)) \]
\[ w \cdot f(p) \leq w \cdot f(opt) \]
\[ w \cdot g(opt) \]

\[ g(p) + w \cdot h(p) \leq w \cdot g(p) + w \cdot h(p) \]
Duplicate Dropping can be Important

Introduction
Algorithm Overview
1: Greedy Phase
2: Cleanup Phase
Empirical Evaluation
Further Observations
Conclusion
Additional Slides
- Loose Bounds
- Duplicates
- Pseudo Code

Four-way Grid Pathfinding (Unit cost)

Nodes generated (relative to A*)

Sub-optimality Bound

wA*

wA* dd
Sometimes it isn’t
Duplicates can be delayed during the greedy search phase.
Optimistic Search\((\text{initial}, \text{bound})\)
1. \(o_{f} \leftarrow \{\text{initial}\}\)
2. \(\hat{o}_{f} \leftarrow \{\text{initial}\}\)
3. incumbent \(\leftarrow \infty\)
4. repeat until \(\text{bound} \cdot f(\text{first on } o_{f}) \geq f(\text{incumbent})\):
5. \(\text{if } \hat{f}(\text{first on } \hat{o}_{f}) < \hat{f}(\text{incumbent}) \text{ then}\)
6. \(n \leftarrow \text{remove first on } o_{f}\)
7. remove \(n\) from \(o_{f}\)
8. else \(n \leftarrow \text{remove first on } o_{f}\)
9. remove \(n\) from \(\hat{o}_{f}\)
10. add \(n\) to \(\text{closed}\)
11. if \(n\) is a goal then
12. incumbent \(\leftarrow n\)
13. else for each child \(c\) of \(n\)
14. \(\text{if } c\text{ is duplicated in } o_{f} \text{ then}\)
15. \(\text{if } c\text{ is better than the duplicate then}\)
16. replace copies in \(o_{f}\) and \(\hat{o}_{f}\)
17. else if \(c\) is duplicated in \(\text{closed}\) then
18. \(\text{if } c\text{ is better than the duplicate then}\)
19. add \(c\) to \(o_{f}\) and \(\hat{o}_{f}\)
20. else add \(c\) to \(o_{f}\) and \(\hat{o}_{f}\)