Planning Algorithms:
When Optimal Is Just Not Good Enough

Wheeler Ruml

University of New Hampshire
Department of Computer Science

Joint work with the UNH AI Group. Support from NSF and DARPA.
Robot Time!

Introduction

■ Robot Time!
■ An Agent
■ Roles

Planning as Search

Optimal Planning

Bounded Suboptimal

Conclusion
The Role of Planning

Introduction
- Robot Time!
- An Agent
- Roles

Planning as Search

Optimal Planning

Bounded Suboptimal

Conclusion
The Role of Planning

- Introduction
- Robot Time!
- An Agent
- Roles

Planning as Search

Optimal Planning

Bounded Suboptimal

Conclusion
The Roles of Planning

- autonomy
  - exploration
  - transportation
  - manufacturing
  - autonomic systems

- decision support
  - operations management
  - personal health
  - eldercare
  - education
Planning as Heuristic Graph Search
Given:

- current state of the world
- models of available actions: \textit{preconditions, effects, costs}
- desired state of the world (partially specified?)

Find:

- cheapest plan
Given:

- start state: an explicit node
- expand function: lazily generate children and their costs
- goal test: predicate on nodes

Find:

- cheapest path to a goal node
Graph Search

Given:
- start state: an explicit node
- expand function: lazily generate children and their costs
- goal test: predicate on nodes

Find:
- cheapest path to a goal node
Graph Search

Given:
- start state: an explicit node
- expand function: lazily generate children and their costs
- goal test: predicate on nodes

Find:
- cheapest path to a goal node
Example: Motion Planning

Introduction
Planning as Search
■ Planning
■ Graph Search
■ Example
■ 3 Algorithms

Optimal Planning
Bounded Suboptimal
Conclusion
1. Uniform Cost Search (Dijkstra, 1959)
3. Explicit Estimation Search (Thayer and Ruml, 2011)
Optimal Planning

Introduction
Planning as Search

Optimal Planning
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Example
- Problem Settings

Bounded Suboptimal

Conclusion
Uniform Cost Search (Dijkstra, 1959)

Introduction
Planning as Search
Optimal Planning
• Uniform Search
  • Dijkstra Behavior
  • Warcraft
  • Knowledge
  • Heuristics
  • Heuristic Search
  • A* Behavior
  • Dijkstra vs A*
  • Lower Bound
  • Problems with A*
  • Example
  • Problem Settings
Bounded Suboptimal
Conclusion
Explore nodes in increasing order of cost-so-far ($g(n)$):
Uniform Cost Search (Dijkstra, 1959)

Explore nodes in increasing order of cost-so-far \( g(n) \):

\[
\text{open} \leftarrow \text{ordered list containing the initial state}
\]

Loop

If \( \text{open} \) is empty, return failure

Node \( \leftarrow \) pop cheapest node off \( \text{open} \)

If Node is a goal, return it (or path to it)

Children \( \leftarrow \) Expand(Node).

Merge Children into \( \text{open} \), keeping sorted by \( g(n) \).
Dijkstra Behavior

- Introduction
- Planning as Search
- Optimal Planning
  - Uniform Search
  - Dijkstra Behavior
  - Warcraft
  - Knowledge
  - Heuristics
  - Heuristic Search
  - A* Behavior
  - Dijkstra vs A*
  - Lower Bound
  - Problems with A*
  - Example
  - Problem Settings
- Bounded Suboptimal
- Conclusion

Wheeler Ruml (UNH)
Dijkstra Behavior

Introduction
Planning as Search
Optimal Planning
  ■ Uniform Search
  ■ Dijkstra Behavior
  ■ Warcraft
  ■ Knowledge
  ■ Heuristics
  ■ Heuristic Search
  ■ A* Behavior
  ■ Dijkstra vs A*
  ■ Lower Bound
  ■ Problems with A*
  ■ Example
  ■ Problem Settings
Bounded Suboptimal
Conclusion

This is not reasonable!

\( g < f^* \)
Dijkstra: Pathfinding in Warcraft

- Introduction
- Planning as Search
- Optimal Planning
  - Uniform Search
  - Dijkstra Behavior
  - Warcraft
    - Knowledge
    - Heuristics
    - Heuristic Search
    - A* Behavior
    - Dijkstra vs A*
    - Lower Bound
    - Problems with A*
    - Example
    - Problem Settings
- Bounded Suboptimal
- Conclusion
This is not reasonable!
Heuristic Estimates of Cost-to-go

Introduction
Planning as Search
Optimal Planning
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
  - Heuristic Search
  - A* Behavior
  - Dijkstra vs A*
  - Lower Bound
  - Problems with A*
  - Example
  - Problem Settings
Bounded Suboptimal
Conclusion

\[ h(n) \leq c^*(n, \text{goal}) \]

'admissible'
Cost should include both cost-so-far and cost-to-go:

\[ g(n) = \text{cost incurred so far} \]
\[ h(n) = \text{lower bound on cost to goal} \]
\[ f(n) = g(n) + h(n) \]
Cost should include both cost-so-far and cost-to-go:

\[
\begin{align*}
g(n) &= \text{cost incurred so far} \\
h(n) &= \text{lower bound on cost to goal} \\
f(n) &= g(n) + h(n)
\end{align*}
\]

*open* ← ordered list containing the initial state

Loop

- If *open* is empty, return failure
- *Node* ← pop cheapest node off *open*
- If *Node* is a goal, return it (or path to it)

*Children* ← Expand(*Node*)

Merge *Children* into *open*, keeping sorted by \(f(n)\)
Cost should include both cost-so-far and cost-to-go:

\[ g(n) = \text{cost incurred so far} \]
\[ h(n) = \text{lower bound on cost to goal} \]
\[ f(n) = g(n) + h(n) \]

\( open \leftarrow \) ordered list containing the initial state

Loop

If \( open \) is empty, return failure

\( Node \leftarrow \) pop cheapest node off \( open \)

If \( Node \) is a goal, return it (or path to it)

\( Children \leftarrow \text{Expand}(Node) \)

Merge \( Children \) into \( open \), keeping sorted by \( f(n) \)

finds optimal solution if heuristic is admissible
A* Behavior

Introduction

Planning as Search

Optimal Planning
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Example
- Problem Settings

Bounded Suboptimal

Conclusion
**A* Behavior**

- **Introduction**
- **Planning as Search**
- **Optimal Planning**
  - Uniform Search
  - Dijkstra Behavior
  - Warcraft
  - Knowledge
  - Heuristics
  - Heuristic Search
  - **A* Behavior**
    - Dijkstra vs A*
    - Lower Bound
    - Problems with A*
    - Example
    - Problem Settings
- **Bounded Suboptimal**
- **Conclusion**
A* Behavior

Introduction
Planning as Search
Optimal Planning
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
- Heuristic Search
- A* Behavior
  - Dijkstra vs A*
  - Lower Bound
  - Problems with A*
  - Example
  - Problem Settings
Bounded Suboptimal
Conclusion

\[ g + h < f^* \]
# Dijkstra vs A*: Pathfinding in Warcraft

<table>
<thead>
<tr>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning as Search</td>
</tr>
<tr>
<td>Optimal Planning</td>
</tr>
<tr>
<td>- Uniform Search</td>
</tr>
<tr>
<td>- Dijkstra Behavior</td>
</tr>
<tr>
<td>- Warcraft</td>
</tr>
<tr>
<td>- Knowledge</td>
</tr>
<tr>
<td>- Heuristics</td>
</tr>
<tr>
<td>- Heuristic Search</td>
</tr>
<tr>
<td>- A* Behavior</td>
</tr>
<tr>
<td>- <strong>Dijkstra vs A</strong>*</td>
</tr>
<tr>
<td>- Lower Bound</td>
</tr>
<tr>
<td>- Problems with A*</td>
</tr>
<tr>
<td>- Example</td>
</tr>
<tr>
<td>- Problem Settings</td>
</tr>
<tr>
<td>Bounded Suboptimal</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Dijkstra vs A*: Pathfinding in Warcraft

Introduction
Planning as Search
Optimal Planning
■ Uniform Search
■ Dijkstra Behavior
■ Warcraft
■ Knowledge
■ Heuristics
■ Heuristic Search
■ A* Behavior
■ Dijkstra vs A*
■ Lower Bound
■ Problems with A*
■ Example
■ Problem Settings
Bounded Suboptimal
Conclusion
Quick Note: A Lower Bound on Solution Cost

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

\( g(n) \) is actual cost-so-far, so when \( h(n) \) is a lower bound on cost-to-go, \( f(n) \) is a lower bound on cost of plan through \( n \)
Quick Note: A Lower Bound on Solution Cost

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

\( g(n) \) is actual cost-so-far, so when \( h(n) \) is a lower bound on cost-to-go, \( f(n) \) is a lower bound on cost of plan through \( n \)

lowest \( f(n) \) on frontier gives lower bound for entire problem!

\[
\text{best}_f = \arg\min_{n \in \text{open}} f(n)
\]
<table>
<thead>
<tr>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning as Search</td>
</tr>
<tr>
<td>Optimal Planning</td>
</tr>
<tr>
<td>■ Uniform Search</td>
</tr>
<tr>
<td>■ Dijkstra Behavior</td>
</tr>
<tr>
<td>■ Warcraft</td>
</tr>
<tr>
<td>■ Knowledge</td>
</tr>
<tr>
<td>■ Heuristics</td>
</tr>
<tr>
<td>■ Heuristic Search</td>
</tr>
<tr>
<td>■ A* Behavior</td>
</tr>
<tr>
<td>■ Dijkstra vs A*</td>
</tr>
<tr>
<td>■ Lower Bound</td>
</tr>
<tr>
<td>Problems with A*</td>
</tr>
<tr>
<td>■ Example</td>
</tr>
<tr>
<td>■ Problem Settings</td>
</tr>
<tr>
<td>Bounded Suboptimal</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
A* takes exponential memory
A* takes exponential memory

Can sometimes be fixed: see ‘iterative deepening’
A* takes exponential memory

Can sometimes be fixed: see ‘iterative deepening’

A* takes exponential time

Helmert and Röger, “How Good is Almost Perfect?” AAAI-08
A* takes exponential memory

Can sometimes be fixed: see ‘iterative deepening’

A* takes exponential time

Helmert and Röger, “How Good is Almost Perfect?” AAAI-08

We must trade cost for time.
Optimizing Utility of Cost + Time

Introduction
Planning as Search
Optimal Planning
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
Example
- Problem Settings
Bounded Suboptimal
Conclusion

15-puzzle

log10 cost/time ratio

log10 factor of best utility

A*
Speedy
Bugsy

A*

Speedy

Bugsy

Wheeler Ruml (UNH)
**Introduction**

Planning as Search

Optimal Planning

- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Heuristics
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Example

---

**Problem Settings**

- **optimal:** minimize solution cost
  must expand all with $f(n) < f^*(opt)$
A New Generation of Problem Settings

optimal: minimize solution cost
must expand all with \( f(n) < f^*(opt) \)

greedy: minimize solving time

anytime: incrementally converge to optimal

bounded suboptimal: minimize time subject to relative cost bound (factor of optimal)

bounded cost: minimize time subject to absolute cost bound

contract: minimize cost subject to absolute time bound

utility function: maximize utility function of cost and time
eg, goal achievement time = plan makespan + search time
A New Generation of Problem Settings

**optimal:** minimize solution cost
must expand all with $f(n) < f^*(opt)$

**greedy:** minimize solving time

**anytime:** incrementally converge to optimal

**bounded suboptimal:** minimize time subject to relative cost bound (factor of optimal)

**bounded cost:** minimize time subject to absolute cost bound

**contract:** minimize cost subject to absolute time bound

**utility function:** maximize utility function of cost and time
eg, goal achievement time = plan makespan + search time
Bounded Suboptimal Search
unbiased estimates can be more informed than lower bounds

nearest goal is the easiest to find
unbiased estimates can be more informed than lower bounds

nearest goal is the easiest to find
unbiased estimates can be more informed than lower bounds

nearest goal is the easiest to find
unbiased estimates can be more informed than lower bounds
nearest goal is the easiest to find

minimize solving time subject to cost $\leq w \cdot \text{optimal}$:

pursue nearest goal estimated to lie within bound

need more information than just lower bound on cost ($h(n)$)!
1. $h$: a lower bound on cost-to-go
\[ f(n) = g(n) + h(n) \]
the traditional optimal A* lower bound
1. $h$: a lower bound on cost-to-go
   $$f(n) = g(n) + h(n)$$
   the traditional optimal A* lower bound

2. $\hat{h}$: an estimate of cost-to-go
   unbiased estimates can be more informed
   $$\hat{f}(n) = g(n) + \hat{h}(n)$$
   (Thayer and Ruml, ICAPS-11)
Three Heuristic Sources of Information

1. $h$: a lower bound on cost-to-go
   \[ f(n) = g(n) + h(n) \]
   the traditional optimal A* lower bound

2. $\hat{h}$: an estimate of cost-to-go
   unbiased estimates can be more informed
   \[ \hat{f}(n) = g(n) + \hat{h}(n) \]
   (Thayer and Ruml, ICAPS-11)

3. $\hat{d}$: an estimate of distance-to-go
   nearest goal is the easiest to find
   (Pearl and Kim, IEEE PAMI 1982, Thayer et al, ICAPS-09)
1. $h$: a lower bound on cost-to-go
   $$ f(n) = g(n) + h(n) $$
   the traditional optimal A* lower bound

2. $\hat{h}$: an estimate of cost-to-go
   unbiased estimates can be more informed
   $$ \hat{f}(n) = g(n) + \hat{h}(n) $$
   (Thayer and Ruml, ICAPS-11)

3. $\hat{d}$: an estimate of distance-to-go
   nearest goal is the easiest to find
   (Pearl and Kim, IEEE PAMI 1982, Thayer et al, ICAPS-09)

   pursue nearest goal estimated to lie within bound
Search Strategy: Which Node to Expand?

\[ \text{best}_f: \text{ open node giving lower bound on cost } \]

\[ \arg\min_{n \in \text{open}} f(n) \]
Search Strategy: Which Node to Expand?

- $best_f$: open node giving lower bound on cost
  \[
  \arg\min_{n \in \text{open}} f(n)
  \]

- $best_{\hat{f}}$: open node giving estimated optimal cost
  \[
  \arg\min_{n \in \text{open}} \hat{f}(n)
  \]
Search Strategy: Which Node to Expand?

\( best_f \): open node giving lower bound on cost

\[
\arg\min_{n \in \text{open}} f(n)
\]

\( best_{\hat{f}} \): open node giving estimated optimal cost

\[
\arg\min_{n \in \text{open}} \hat{f}(n)
\]

pursue nearest goal estimated to lie within bound

\( best_{\hat{d}} \): estimated \( w \)-suboptimal node with minimum \( \hat{d} \)

\[
\arg\min_{n \in \text{open} \wedge \hat{f}(n) \leq w \cdot \hat{f}(best_{\hat{f}})} \hat{d}(n)
\]
Introduction
Planning as Search
Optimal Planning

Bounded Suboptimal
- Intuition
- Three Heuristics
- Strategy
- Expansion Order
- Bound
- EES Performance

Conclusion

---

\( \text{best}_f \): open node giving lower bound on cost
\( \text{best}_{\hat{f}} \): open node giving estimated optimal cost
\( \text{best}_{\hat{d}} \): estimated \( w \)-suboptimal node with minimum \( \hat{d} \)

node to expand next:

1. pursue the nearest goal estimated to lie within the bound
2.
3.

in other words:

1. \( \text{best}_{\hat{d}} \)
2.
3.
**EES Expansion Order**

- **best_f**: open node giving lower bound on cost
- **best_\hat{f}**: open node giving estimated optimal cost
- **best_\hat{d}**: estimated \( w \)-suboptimal node with minimum \( \hat{d} \)

Node to expand next:
1. pursue the nearest goal estimated to lie within the bound
2.
3.

In other words:
1. \( \text{if} \ \hat{f}(\text{best}_\hat{d}) \leq w \cdot f(\text{best}_f) \ \text{then} \ \text{best}_\hat{d} \)
2.
3.
best_f: open node giving lower bound on cost

best_f̂: open node giving estimated optimal cost

best_d̂: estimated w-suboptimal node with minimum d̂

Node to expand next:
1. pursue the nearest goal estimated to lie within the bound
2. pursue the estimated optimal solution
3. in other words:

1. if \( \hat{f}(\text{best}_d) \leq w \cdot f(\text{best}_f) \) then \( \text{best}_d \)
2. else if \( \hat{f}(\text{best}_f) \leq w \cdot f(\text{best}_f) \) then \( \text{best}_f \)
3.
\( \text{best}_f \): open node giving lower bound on cost
\( \text{best}_\hat{f} \): open node giving estimated optimal cost
\( \text{best}_\hat{d} \): estimated \( w \)-suboptimal node with minimum \( \hat{d} \)

node to expand next:
1. pursue the nearest goal estimated to lie within the bound
2. pursue the estimated optimal solution
3. raise the lower bound on optimal solution cost

in other words:
1. \( \text{if } \hat{f}(\text{best}_\hat{d}) \leq w \cdot f(\text{best}_f) \text{ then } \text{best}_\hat{d} \)
2. \( \text{else if } \hat{f}(\text{best}_\hat{f}) \leq w \cdot f(\text{best}_f) \text{ then } \text{best}_\hat{f} \)
3. \( \text{else } \text{best}_f \)

see paper for further justification. Note: no magic numbers!
how does $\hat{f}(n) \leq w \cdot f(best_f)$ ensure the suboptimality bound?
how does $\hat{f}(n) \leq w \cdot f(best_f)$ ensure the suboptimality bound?

\[
\begin{align*}
    f(n) & \leq \hat{f}(n) & f(n) \text{ is a lower bound for } n \\
    \hat{f}(n) & \leq w \cdot f(best_f) & \text{expansion criterion} \\
    w \cdot f(best_f) & \leq w \cdot f^*(opt) & \text{because } f(best_f) \text{ is a lower bound for the entire problem} \\
    f(n) & \leq w \cdot f^*(opt) & \text{suboptimality bound}
\end{align*}
\]
bounded suboptimal search:
minimize time subject to
relative cost bound (factor of optimal)
Dock Robot

Suboptimality

total raw cpu time

Suboptimality

A* eps

wA*

Optimistic

Skeptical

EES

EES Opt.
EES Performance

Vacuum World

log10 total raw cpu time

Suboptimality

A* eps
Optimistic
Skeptical
wA*
EES
EES Opt.

Wheeler Ruml (UNH) When Optimal Is Just Not Good Enough – 29 / 34
EES Performance

Introduction
Planning as Search
Optimal Planning
Bounded Suboptimal
■ Intuition
■ Three Heuristics
■ Strategy
■ Expansion Order
■ Bound
■ EES Performance
Conclusion

Heavy Vacuum World

log10 total raw cpu time

Suboptimality

wA*
Optimistic
Skeptical
A* eps
EES
EES Opt.
Conclusion
A Science of Suboptimal Search

- what are the problem settings?
  - guarantees beyond optimal search
  - eg, bounded suboptimal search
  - utility-based optimization

- what are the sources of information?
  - where do inadmissible heuristics come from?
  - estimates instead of lower bounds
  - distance in addition to cost

- how to exploit and combine information?
  - new generation of suboptimal algorithms
  - meta-reasoning
Search Algorithms as Agents

Introduction
Planning as Search
Optimal Planning
Bounded Suboptimal
Conclusion
- Subopt. Search
- The AI Vision
- Algs as Agents

Wheeler Ruml (UNH)
The University of New Hampshire