Planning Algorithms: When Optimal Just Isn’t Good Enough

Wheeler Ruml

University of New Hampshire
Department of Computer Science

Joint work with the UNH AI Group. Funded by NSF and DARPA.
Where are the Walking Talking Robots?

Introduction

■ Robots?
■ Toddler Steps
■ An Agent

Planning as Search

Bounded Suboptimal

Conclusion

Lt. Commander Data
**Star Trek: TNG**

Lt. Sharon Valerii
**Battlestar Galactica**

Zoe Greystone
**Caprica**
Toddler Steps

Introduction

Robots?

Toddler Steps

An Agent

Planning as Search

Bounded Suboptimal

Conclusion
The Role of Planning

Introduction
- Robots?
- Toddler Steps
- An Agent

Planning as Search

Bounded Suboptimal

Conclusion
The Role of Planning

Introduction
- Robots?
- Toddler Steps
- An Agent

Planning as Search

Bounded Suboptimal

Conclusion
Planning as Heuristic Graph Search
Given:
- current state of the world
- models of available actions
  - preconditions, effects, costs
- desired state of the world (partially specified?)

Find:
- cheapest plan
Graph Search

Given:
- start state: an explicit node
- expand function: generates children and their costs
  \( \text{assume arc cost} \geq 0 \)
- goal test: predicate on nodes

Find:
- cheapest path to a goal node
Graph Search

Given:
- start state: an explicit node
- expand function: generates children and their costs
  
  \[\text{assume arc cost } \geq 0\]
- goal test: predicate on nodes

Find:
- cheapest path to a goal node
Given:
- start state: an explicit node
- expand function: generates children and their costs
  \( \text{assume arc cost} \geq 0 \)
- goal test: predicate on nodes

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

Introduction
Planning as Search
- Planning
- Graph Search
- Example
  - The Problem
  - 3 Algorithms
  - Uniform Search
  - Dijkstra Behavior
  - Warcraft
  - Knowledge
  - Lower Bounds
  - Heuristic Search
  - A* Behavior
  - Dijkstra vs A*
  - Lower Bound
  - Problems with A*
  - Problem Settings

Bounded Suboptimal

Conclusion

Wheeler Ruml (UNH)
Ever wonder why no one draws trees more than 3 levels deep?
Ever wonder why no one draws trees more than 3 levels deep?

\[
14^{100} > 10^{80}
\]
A Tale of Three Algorithms

1. Uniform Cost Search (Dijkstra, 1959)
3. Explicit Estimation Search (Thayer and Ruml, 2011)
Uniform Cost Search (Dijkstra, 1959)

Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- **Uniform Search**
  - Dijkstra Behavior
  - Warcraft
  - Knowledge
  - Lower Bounds
  - Heuristic Search
  - A* Behavior
  - Dijkstra vs A*
  - Lower Bound
  - Problems with A*
  - Problem Settings

Bounded Suboptimal

Conclusion
Uniform Cost Search (Dijkstra, 1959)

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 \(g(n)\):

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

Loop

If \(Q\) is empty, return failure

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

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

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

Merge \(Children\) into \(Q\), keeping sorted by \(g(n)\).
Dijkstra Behavior

Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search

Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion
Dijkstra Behavior

Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search

Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion
Dijkstra Behavior

Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search

Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion

This is not reasonable!
Dijkstra: Pathfinding in Warcraft

Introduction
Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior

Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion
Problem-Specific Background Knowledge

Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft

Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion

This is not reasonable!
Lower Bounds on Cost-to-go

\[ h(n) < f^*(n) \]
Cost should include both cost-so-far and cost-to-go:
Cost should include both cost-so-far and cost-to-go:

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

**Loop**

- If \( Q \) is empty, return failure
- \( Node \leftarrow \text{pop cheapest node off } Q \)
- If \( Node \) is a goal, return it (or path to it)
- \( Children \leftarrow \text{Expand}(Node) \)
- Merge \( Children \) into \( Q \), keeping sorted by \( f(n) \)

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

Introduction
Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal
Conclusion
Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search

A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal

Conclusion
A* Behavior

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

Introduction
Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- Problem Settings

Bounded Suboptimal
Conclusion
### Introduction

Planning as Search
- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound
- Problems with A*
- 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 final solution cost.

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

$$\text{best}_f = \arg\min_{n \in \text{open}} f(n)$$
Problems with A*

Introduction

Planning as Search

- Planning
- Graph Search
- Example
- The Problem
- 3 Algorithms
- Uniform Search
- Dijkstra Behavior
- Warcraft
- Knowledge
- Lower Bounds
- Heuristic Search
- A* Behavior
- Dijkstra vs A*
- Lower Bound

Problems with A*

Problem Settings

Bounded Suboptimal

Conclusion
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
A* takes exponential memory

Can sometimes be fixed: see ‘iterative deepening’

A* takes exponential time

We must trade cost for time.
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

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

**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

- Intuition
- Three Heuristics
- EES
- Expansion Order
- Bound
- EES Performance

Conclusion
- 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

The graph shows the relationship between the size of the problem and the total raw CPU time for admission and inadmissible heuristics in the context of Heavy Vacuum. The graph indicates that as the size of the problem increases, the total raw CPU time also increases significantly for both admission and inadmissible heuristics.
unbiased estimates can be more informed than lower bounds

nearest goal is the easiest to find
The Intuition Behind EES (Thayer and Ruml, 2011)

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

Explicit Estimation Search (EES) combines these ideas

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

pursue the nearest goal within the 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
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)
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)
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)

   pursue the nearest goal within the bound
Finding $\text{best}_f$

$best_f$: open node giving lower bound on cost

$$\arg\min_{n \in \text{open}} f(n)$$
Finding $\text{best}_d$

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

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

$\text{best}_f$: open node giving estimated optimal cost

$$\arg\min_{n \in \text{open}} \hat{f}(n)$$
**Finding \( \text{best}_d \)**

**Introduction**

Planning as Search

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

**Conclusion**

\( \text{best}_f \): open node giving lower bound on cost

\[
\text{argmin}_{n \in \text{open}} f(n)
\]

\( \text{best}_{\hat{f}} \): open node giving estimated optimal cost

\[
\text{argmin}_{n \in \text{open}} \hat{f}(n)
\]

pursue the nearest goal within the bound

\( \text{best}_{\hat{d}} \): estimated \( w \)-suboptimal node with minimum \( \hat{d} \)

\[
\text{argmin}_{n \in \text{open} \land \hat{f}(n) \leq w \cdot \hat{f}(\text{best}_{\hat{f}})} \hat{d}(n)
\]
**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 shortest solution that is within the bound
2. 
3. 

in other words:

1. $best_{\hat{d}}$
2. 
3.
Introduction
Planning as Search
Bounded Suboptimal
  ■ Intuition
  ■ Three Heuristics
  ■ EES
  ■ Expansion Order
  ■ Bound
  ■ EES Performance
Conclusion

**EES Expansion Order**

*best_{\hat{f}}*: open node giving lower bound on cost

*best_{\hat{\hat{d}}}: open node giving estimated optimal cost

*best_{\hat{d}}*: estimated $w$-suboptimal node with minimum $\hat{\hat{d}}$

node to expand next:

1. pursue the shortest solution that is within the bound
2. 
3.

in other words:

1. **if** $\hat{f}(best_{\hat{d}}) \leq w \cdot f(best_{\hat{f}})$ **then** $best_{\hat{d}}$
2.
3.
best_f: open node giving lower bound on cost
best_f: open node giving estimated optimal cost
best_d: estimated w-suboptimal node with minimum \( \hat{d} \)

node to expand next:
1. pursue the shortest solution that is 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.
best_f: open node giving lower bound on cost
best_f: open node giving estimated optimal cost
best_d: estimated w-suboptimal node with minimum \( \hat{d} \)

node to expand next:
1. pursue the shortest solution that is within the bound
2. pursue the estimated optimal solution
3. raise the lower bound on optimal solution cost

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

see paper for further justification
how does \( \hat{f}(n) \leq w \cdot f(\text{best}_f) \) ensure the suboptimality bound?
EES Respects 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) \quad \text{\(f(n)\) is a lower bound for \(n\)} \\
  \hat{f}(n) & \leq w \cdot f(best_f) \quad \text{expansion criterion} \\
  w \cdot f(best_f) & \leq w \cdot f^*(opt) \quad \text{because \(f(best_f)\) is a lower bound for the entire problem} \\
  f(n) & \leq w \cdot f^*(opt) \quad \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

- A* eps
- Optimistic
- Skeptical
- wA*
- EES
- EES Opt.

log10 total raw cpu time

Suboptimality

0.6
0
-0.6

Wheeler Ruml (UNH)

When Optimal Just Isn’t Good Enough – 28 / 33
Heavy Vacuum World

log10 total raw cpu time vs. suboptimality

- wA*
- Optimistic
- Skeptical
- A* eps
- EES
- EES Opt.
## Conclusion

- **Summary**
- **The AI Vision**
- **Algs as Agents**
- **UNH**
Who’s afraid of big bad NP-hardness?

**bounded suboptimal** planning

- fast solving
- control over cost

a new generation of planning algorithms

- beyond optimal
- estimates, in addition to lower bounds
- new sources of heuristic information
- principled, provable performance
The AI Vision

Introducing the concept of planning as search, the diagram illustrates the interaction between an agent, its world model, planner, and search mechanisms. The agent interacts with the world through sensing and actions, with the planner facilitating the process through a world model and search algorithms.

Summary

- **The AI Vision**
- **Algs as Agents**
- **UNH**

Wheeler Ruml (UNH)
Search Algorithms as Agents

Introduction
Planning as Search
Bounded Suboptimal
Conclusion

- Summary
- The AI Vision
- Algs as Agents
- UNH

Search algorithm

model of space

meta-reasoner

open list

domain

expand

children, g, h values
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, bioinformatics