Faster Bounded-Cost Search Using Inadmissible Estimates

Jordan Thayer, Roni Stern, Ariel Felner, Wheeler Ruml

jthayer@sift.net  roni.stern@gmail.com  felner@bgu.ac.il  ruml@cs.unh.edu
Motivation: Optimal Search Hard, Greedy Solutions Expensive

optimal search won’t scale
greedy solutions too expensive
Motivation: Bounded Suboptimal Provides Middle Ground

- **Introduction**
- **Motivation**
- **Outline**
- **Potential Search**
- **BEES**
- **Conclusion**
- **BEEPS, PTS-\(\hat{h}\)**

**Grid Four-way 35%**

**Total nodes generated**

- A*
- wA*
- greedy

**Final sol cost relative to A**

- greedy
- wA*
- A*

impose bounds to limit cost (relative to optimal)
Motivation: What if I have a Budget instead of $w$?

What if we have a budget instead of a relative bound?

**Bounded Cost Search**: find a solution of cost no more than $C$
motivation

previous approach: potential search

problem with potential search

BEES

bigger picture

PTS on inadmissible heuristics, BEEPS
best-first search in order of chance of satisfying cost bound $C$
best-first search in order of chance of satisfying cost bound $C$

$$PT_C(n) = Pr(g(n) + h^*(n) \leq C)$$
Potential Search

Introduction

Potential Search

PTS

Performance

Shortcoming

BEES

Conclusion

BEEPS, PTS-\(\hat{h}\)

best-first search in order of chance of satisfying cost bound \(C\)

\[ P_{T_C}(n) = Pr(g(n) + h^*(n) \leq C) \]

unfortunately, we may not be able to compute that

\[ f_{lnr}(n) = \frac{h(n)}{C - g(n)} \]

produces an equivalent order under certain assumptions
PTS Performance: Non-Unit Cost Performance is Bad!

Introduction

Potential Search
- PTS
- Performance
- Shortcoming

BEES

Conclusion

BEEPS, PTS-$h$

100 Inverse 15 Puzzles

log10 total raw cpu time

Cost Bound

PTS

Speedy w/ Cutoff
Potential Search Ignores Solution Length

\[ d(A) = 6, \quad h(A) = 6 \]
\[ d(T) = 1, \quad h(T) = 10 \]
\[ C = 20 \]
Potential Search Ignores Solution Length

\[ d(A) = 6, \quad h(A) = 6 \]
\[ d(T) = 1, \quad h(T) = 10 \]
\[ C = 20 \]

\[ f_{lnr}(A) = \frac{6}{20-1} \]
\[ f_{lnr}(T) = \frac{10}{20-10} \]
Potential Search Ignores Solution Length

\[ d(A) = 6, \quad h(A) = 6 \]
\[ d(T) = 1, \quad h(T) = 10 \]
\[ C = 20 \]

\[ f_{lnr}(A) = \frac{6}{20 - 1} \]
\[ f_{lnr}(T) = \frac{10}{20 - 10} \]

PTS prefers A to T
Bounded Cost Explicit Estimation Search
1. avoid reliance on error models (ie $f_{lnr}$)

2. improve performance on non-unit cost domains without losing performance in unit cost domains
1. estimate which nodes are within cost bound
2. best-first search of these on estimated actions-to-go
Step 1: Estimate which nodes are within cost bound

\( h(n) \): an admissible cost-to-go estimate

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

\( f(n) \leq C \): could lead to a solution in bound

\[
focal = \{ n \in open \mid f(n) \leq C \}
\]
Step 1: Estimate which nodes are within cost bound

\[ h(n) \text{: an admissible cost-to-go estimate} \]
\[ \hat{h}(n) \text{: best-guess estimate of cost-to-go} \]

\[ f(n) = g(n) + h(n) \]
\[ \hat{f}(n) = g(n) + \hat{h}(n) \]

\[ f(n) \leq C \text{: could lead to a solution in bound} \]
\[ \hat{f}(n) \leq C \text{: probably leads to a solution in bound} \]

\[ \text{focal} = \{ n \in \text{open} | \hat{f}(n) \leq C \} \]
What If No Nodes Appear to Be Within Bound?

\[ \hat{f}(n) = g(n) + \hat{h}(n) \]
\[ f(n) = g(n) + h(n) \]
\[ \hat{f}(n) \geq f(n) \]

\[
\text{focal} = \{ n \in \text{open} | \hat{f}(n) \leq C \} \\
\text{open} = \{ n | f(n) \leq C \}
\]

\( A^* \) provides an efficient way to prove no solution in \( C \)
What If No Nodes Appear to Be Within Bound?

\[
\begin{align*}
\hat{f}(n) &= g(n) + \hat{h}(n) \\
 f(n) &= g(n) + h(n) \\
\hat{f}(n) &\geq f(n)
\end{align*}
\]

\[
\text{focal} = \{ n \in \text{open} \mid \hat{f}(n) \leq C \}
\]

\[
\text{open} = \{ n \mid f(n) \leq C \}
\]

**A**\(^*\) provides an efficient way to prove no solution in \(C\)

1. Estimate which nodes are within cost bound
2. Best-first search of these on estimated actions-to-go
3. **A**\(^*\) search if we think no solution exists within \(C\)
BEES is a best first search on the following rule

\[
\text{open} = \{ n | f(n) \leq C \}
\]

\[
\text{focal} = \{ n \in \text{open} | \hat{f}(n) \leq C \}
\]

**selectNode**

1. **if** focal \( \neq \{ \} \)
2. **then** return \( n \in \text{focal} \) estimated nearest to a goal
3. **else** return \( n \in \text{open} \) with minimum \( f \)
PTS vs. BEES on Explicit Graph

BEES
- Goals
- Algorithm
- Defining Focal
- Pseudo-Code

Example

Thayer et al

C = 20  \[ f_{lnr}(A) = \frac{6}{20-1} \]

\[ f_{lnr}(T) = \frac{10}{20-10} \]

BEES prefers T, PTS prefers A
Empirical Evaluation

Introduction

Potential Search

BEES
- Goals
- Algorithm
- Defining Focal
- Pseudo-Code
- Example
- Performance

Conclusion

BEEPS, PTS-\(\hat{h}\)

Korf's 100 15 Puzzles

\[
\text{log10 total raw cpu time}
\]

Cost Bound

Speedy w/ Cutoff

BEES

PTS
Empirical Evaluation

Introduction

Potential Search

BEES
- Goals
- Algorithm
- Defining Focal
- Pseudo-Code
- Example

Performance

Conclusion

BEEPS, PTS-$\hat{h}$

Thayer et al
BEES outperforms previous state-of-the-art when action costs differ, take advantage of $d$
inadmissible heuristics can speed up search
finding solutions, making proofs are different
Finding Solutions, Proving Bounds Different Tasks

Introduction

Potential Search

BEES

Conclusion

Summary

BEEPS, PTS–$\hat{h}$

---

**Life Four-way Grid World**

![Graph showing final solution cost relative to A* compared to suboptimality bound.](image)

- $y=x$
- $A^*$ epsilon
- EES
- $wA^*$
- RD$wA^*$
- Clamped Adaptive
- Dw$A^*$
Inadmissible Heuristics Speed Search

- **Introduction**
- **Potential Search**
- **BEES**
- **Conclusion**
- **Summary**

**Dock Robot**

- $wA^*$
- Optimistic
- Skeptical

![Graph showing performance with suboptimality and log10 total raw CPU time for different heuristics.](image)
Use $d$ to Go Fast

Korf’s 100 15 Puzzles - Inverse Cost

Solution Quality

log10 total raw cpu time

greedy on $h$

greedy on $d$

Thayer et al
<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ BEES outperforms previous state-of-the-art</td>
</tr>
<tr>
<td>■ when action costs differ, take advantage of $d$</td>
</tr>
<tr>
<td>■ inadmissible heuristics can speed up search</td>
</tr>
<tr>
<td>■ finding solutions, making proofs are different</td>
</tr>
</tbody>
</table>
Using $\hat{h}$ in PTS

\[ f_{lnr}(n) = \frac{h(n)}{C - g(n)} \]

\[ f_{lnr}(n) = \frac{\hat{h}(n)}{C - g(n)} \]
BEES is a best first search on the following rule

selectNode
1. if focal $\neq \{\}$
2. then return $n \in \text{focal}$ with minimum $\hat{d}$
3. else return $n \in \text{open}$ with minimum $f_{lnr}$
Empirical Results

Heavy Vacuum World

log10 total raw cpu time

PTS
PTS - h\(^\wedge\)
BEEPS
BEES

Cost Bound

Using \(h\) in PTS
BEEPS - BEES
with Potential Measurements

Performance
EES as BC

Thayer et al
Empirical Results

Introduction

Potential Search

BEES

Conclusion

BEEPS, PTS-\(\hat{h}\)

- Using \(\hat{h}\) in PTS
- BEEPS - BEES with Potential Measurements

Performance

EES as BC

100 Inverse 15 Puzzles

\(\log_{10}\) total raw cpu time

Cost Bound

 PTS
PTS - \(\hat{h}\)
BEEPS
BEES

Thayer et al
Empirical Results

- Introduction
- Potential Search
- BEES
- Conclusion

BEEPS, PTS-\( \hat{h} \)
- Using \( \hat{h} \) in PTS
- BEEPS - BEES with Potential Measurements
- Performance
- EES as BC

Thayer et al
EES Doesn’t Work as Bounded Cost Algorithm

Introduction

Potential Search

BEES

Conclusion

BEEPS, PTS-h

- Using \( h \) in PTS
- BEEPS - BEES with Potential Measurements
- Performance
- EES as BC

100 Inverse 15 Puzzles

log10 total raw cpu time

Cost Bound

PTS

EES w=1.5

PTS - h

EES w=3

EES w=5

BEEPS

BEES

Thayer et al