Search Algorithms as Agents
—or—
There’s More to Life Than $h(n)$

Wheeler Ruml

and the UNH AI Group, esp. Jordan Thayer

(thanks to the NSF RI and DARPA CSSG programs for support)
The AI Vision

- Heuristic Search
- Alg as Agent
- What is Search?
- Related Work
- Problem Settings

Greedy Search
Bounded Search
Contract Search
Utility Functions
Conclusion

- World model
- Planner
- Search
- Sensing
- Actions
- World
Search Algorithms as Agents

Heuristic Search
- The AI Vision
- Alg as Agent
- What is Search?
- Related Work
- Problem Settings

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

search algorithm

domain

search strategy

open list

children, g, h values

expand

Wheeler Ruml (UNH)
What is Search?

acting under uncertainty to maximize utility
What is Search?

acting under uncertainty to maximize utility

= all of AI
What is Search?

acting under uncertainty to maximize utility

= all of AI

possible sources of information, ‘sensors’
how to exploit information, ‘aggregation’, ‘filtering’
Rational search:
- Othar Hansson
- Andy Mayer

Metareasoning, ‘bounded optimality’:
- Eric Wefald
- Shlomo Zilberstein
- Eric Horvitz
- Stuart Russell

Anytime search:
- Tom Dean
- Mark Boddy
**optimal:** minimize solution cost
must expand all with $f(n) < f^*(opt)$
Problem Settings

Heuristic Search

■ The AI Vision
■ Alg as Agent
■ What is Search?
■ Related Work
■ Problem Settings

Greedy Search
Bounded Search
Contract Search
Utility Functions
Conclusion

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
Greedy Search

- Direct Approach
- Distance-to-Go
- Speedy Results

Bounded Search

Contract Search

Utility Functions

Conclusion
how to minimize solving time?
how to minimize solving time?
how to minimize number of expansions?
how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
for domains with costs, this is **not** $h(n)$
how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
for domains with costs, this is not $h(n)$

source #1 of 5: distance-to-go
Distance-to-Go Estimates

Heuristic Search
- Direct Approach
- Distance-to-Go
- Speedy Results

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

\[ h = 4 \quad d = 2 \]
\[ h = 5 \quad d = 1 \]
Distance-to-Go Estimates

\[ h = 4 \quad d = 2 \]
\[ h = 5 \quad d = 1 \]

\[ d_{\text{nearest}}(n) = 2 \]
\[ d^{\ast}_{\text{nearest}}(n) = 2 \]

\[ d^{\ast}_{\text{cheapest}}(n) = 3 \]

\( d_{\text{nearest}} \) is potentially independent of \( h \),
can be recomputed as \( h \) with unit costs

\( d_{\text{cheapest}} \) often computable alongside \( h \)

\( d \) can be inadmissible
Distance-to-Go Estimates

Heuristic Search
- Direct Approach
- Distance-to-Go
- Speedy Results

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

\[ h = 4 \quad d = 2 \]
\[ h = 5 \quad d = 1 \]

\[ d_{\text{cheapest}}^*(n) = 3 \]
\[ d_{\text{nearest}}^*(n) = 2 \]

\(d_{\text{nearest}}\) is potentially independent of \(h\), can be computed as \(h\) with unit costs

\(d_{\text{cheapest}}\) often computable alongside \(h\)

\(d\) can be inadmissible

Speedy Search: best-first search on \(d(n)\) (Thayer et al, SoCS-09)
Speedy Results

Life Four-way Grids 35% Obstacles

speedy finds solutions faster
Speedy Results

- Heuristic Search
- Greedy Search
  - Direct Approach
  - Distance-to-Go
  - Speedy Results
- Bounded Search
- Contract Search
- Utility Functions
- Conclusion

Heavey Vacuum

- greedy on h
- greedy on d

Speedy scales better

Wheeler Rum (UNH)
Bounded-Suboptimal Search
minimize solving time subject to relative cost bound (factor of optimal)
minimize solving time subject to relative cost bound (factor of optimal)
weighted $A^*(f'(n) = g(n) + w \cdot h(n))$ is simple but ad hoc
minimize solving time subject to relative cost bound (factor of optimal)
weighted $A^*(f'(n) = g(n) + w \cdot h(n))$ is simple but ad hoc

expand the node closest to a solution within the bound
minimize solving time subject to relative cost bound (factor of optimal)
weighted A* \( f'(n) = g(n) + w \cdot h(n) \) is simple but ad hoc

expand the node closest to a solution within the bound
known to not work!
### A* (Pearl and Kim, IEEE PAMI 1982)

- **A* (Pearl and Kim, IEEE PAMI 1982)**

<table>
<thead>
<tr>
<th>Heuristic Search</th>
<th>Greedy Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounded Search</td>
<td></td>
</tr>
<tr>
<td>Direct Approach</td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>A* 's Flaw</td>
<td></td>
</tr>
<tr>
<td>Estimated Cost</td>
<td>Estimated Cost</td>
</tr>
<tr>
<td>Learning Cost</td>
<td>Learning Cost</td>
</tr>
<tr>
<td>EES</td>
<td>EES</td>
</tr>
<tr>
<td>EES Order</td>
<td>EES Order</td>
</tr>
<tr>
<td>EES Results</td>
<td>EES Results</td>
</tr>
<tr>
<td>The Story So Far</td>
<td>The Story So Far</td>
</tr>
</tbody>
</table>

**intuition:** of all solutions within the bound, the nearest should be the fastest to find
Heuristic Search

Greedy Search

Bounded Search

Direct Approach

\( A^*_\epsilon \)

\( A^*_\epsilon \)'s Flaw

Estimated Cost

Learning Cost

EES

EES Order

EES Results

The Story So Far

Contract Search

Utility Functions

Conclusion

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

\( best_f \): open node with minimum \( f \)

two lists:

\( open \): as usual, sorted on \( f(n) \)

\( focal \): subset of open with \( f(n) \leq w \cdot f(best_f) \), sorted on \( d(n) \)

\( A^*_\epsilon \): best-first search using \( focal \)
intuition: of all solutions within the bound, the nearest should be the fastest to find

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

\( best_f \): open node with minimum \( f \)

two lists:

- \( open \): as usual, sorted on \( f(n) \)
- \( focal \): subset of open with \( f(n) \leq w \cdot f(best_f) \), sorted on \( d(n) \)

\( A^*_\epsilon \): best-first search using \( focal \)

Why doesn’t it work?
**A_ε’s Flaw (Thayer et al, SoCS-09)**

- **Heuristic Search**
  - Greedy Search
  - Bounded Search
    - Direct Approach
    - A_ε
    - A_ε’s Flaw
      - Estimated Cost
      - Learning Cost
      - EES
      - EES Order
      - EES Results
      - The Story So Far

- **Contract Search**
- **Utility Functions**
- **Conclusion**

---

**open**: as usual, sorted on \( f(n) \)

**focal**: subset of open with \( f(n) \leq w \cdot f(best_f) \), sorted on \( d(n) \)

---

\[
\text{open: } \bigcirc \bigcirc \bigcirc \bigcirc \bigcirc \bigcirc
\]

\[
\text{focal: } \bigcirc \bigcirc \bigcirc
\]

---

\( f \) often \( d \)

\( d \) often \( f \)
open: as usual, sorted on $f(n)$
focal: subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$
**A⁺’s Flaw (Thayer et al, SoCS-09)**

**open:** as usual, sorted on $f(n)$

**focal:** subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$

- $f$ rises as search progresses ($h$ is admissible)
- $best_d$’s children don’t qualify for focal
intuition: pursuing the shortest solution within the bound should be fast

intuition': an unbiased estimates of cost won't always rise
intuition: pursuing the shortest solution within the bound should be fast

intuition': an unbiased estimates of cost won’t always rise

source #2 of 5: unbiased cost estimates $\hat{f}(n)$
Learning Unbiased Cost Estimates

Every expansion gives evidence for heuristic’s error!

\[
f^*(p) = f^*(bc)
\]
\[
h^*(p) = h^*(bc) + c(p, bc)
\]
\[
\epsilon_h = (h(bc) + c(p, bc)) - h(p)
\]
Every expansion gives evidence for heuristic’s error!

\[
\begin{align*}
    f^*(p) &= f^*(bc) \\
    h^*(p) &= h^*(bc) + c(p, bc) \\
    \epsilon_h &= (h(bc) + c(p, bc)) - h(p) \\
    \hat{h}(n) &= h(n) + \overline{\epsilon_h} \cdot d(n)
\end{align*}
\]
Learning Unbiased Cost Estimates

Every expansion gives evidence for heuristic’s error!

\[ f^*(p) = f^*(bc) \]
\[ h^*(p) = h^*(bc) + c(p, bc) \]
\[ \epsilon_h = (h(bc) + c(p, bc)) - h(p) \]

\[ \hat{h}(n) = h(n) + \epsilon_h \cdot d(n) \]

can do this for \( d(n) \) too...

\[ \hat{h}(n) = h(n) + \epsilon_h \cdot \hat{d}(n) \]

see Thayer et al (ICAPS-11) for details
Explicit Estimation Search (EES, IJCAI-11)

\[ \text{best}_f : \text{ open node with minimum } f \]

\[ \text{best}_{\hat{f}} : \text{ open node with minimum } \hat{f} \]
explicit estimation search (EES, IJCAI-11)

\(\text{best}_f\): open node with minimum \(f\)

\(\text{best}_{\hat{f}}\): open node with minimum \(\hat{f}\)

three lists (!):

- \textit{open}: as usual, but sorted on \(\hat{f}(n)\)
- \textit{focal}: subset of open with \(\hat{f}(n) \leq w \cdot \hat{f}(\text{best}_{\hat{f}})\), sorted on \(\hat{d}(n)\)
- \textit{cleanup}: same as open, but sorted on \(f(n)\)
Explicit Estimation Search (EES, IJCAI-11)

\[ \text{best}_f: \text{open node with minimum } f \]
\[ \text{best}_{\hat{f}}: \text{open node with minimum } \hat{f} \]

three lists (!):

- **open**: as usual, but sorted on \( \hat{f}(n) \)
- **focal**: subset of open with \( \hat{f}(n) \leq w \cdot \hat{f} (\text{best}_{\hat{f}}) \), sorted on \( \hat{d}(n) \)
- **cleanup**: same as open, but sorted on \( f(n) \)

\[ \text{best}_{\hat{d}}: \text{first node on focal} \]
Explicit Estimation Search (EES, IJCAI-11)

**Heuristic Search**
- Greedy Search
- Bounded Search
  - Direct Approach
  - $A^*_\epsilon$
  - $A^*_\epsilon$’s Flaw
  - Estimated Cost
  - Learning Cost
- EES
  - EES Order
  - EES Results
  - The Story So Far

**Contract Search**
- Utility Functions
- Conclusion

---

$best_f$: open node with minimum $f$

$best_{\hat{f}}$: open node with minimum $\hat{f}$

three lists (!):

- **open**: as usual, but sorted on $\hat{f}(n)$
- **focal**: subset of open with $\hat{f}(n) \leq w \cdot \hat{f}(best_{\hat{f}})$, sorted on $\hat{d}(n)$
- **cleanup**: same as **open**, but sorted on $f(n)$

$best_{\hat{d}}$: first node on **focal**

estimated $w$-admissible node with minimum $\hat{d}$
**EES Expansion Order**

- $best_f$: open node with minimum $f$
- $best_{\hat{f}}$: open node with minimum $\hat{f}$
- $best_{\hat{d}}$: estimated $w$-admissible node with minimum $\hat{d}$

Node to expand next:

1. pursue the shortest solution within the bound
2. pursue the estimated cheapest solution
3. raise the lower bound on optimal cost

In other words:

1. \textbf{if} $\hat{f}(best_{\hat{d}}) \leq w \cdot f(best_f)$ \textbf{then} $best_{\hat{d}}$
2. \textbf{else if} $\hat{f}(best_{\hat{f}}) \leq w \cdot f(best_f)$ \textbf{then} $best_{\hat{f}}$
3. \textbf{else} $best_f$

find a solution provably within the bound as quickly as possible
**Heuristic Search**
- Greedy Search
- Bounded Search
  - Direct Approach
  - $A^*_\epsilon$
  - $A^*_\epsilon$'s Flaw
  - Estimated Cost
  - Learning Cost
  - EES
  - EES Order
- EES Results
- The Story So Far

**Dock Robot**

![Graph showing total raw cpu time vs. suboptimality for different algorithms.](image)

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

Vacuum World

A* eps
Optimistic
Skeptical
wA*
EES
EES Opt.

log10 total raw cpu time

Suboptimality

Wheeler Ruml (UNH)

Search Algorithms as Agents – 22 / 40
search algorithms as agents

more information sources

1. $d(n)$: distance-to-go

2. $\hat{f}(n)$: expected cost (expansion experience)

more problem settings

- greedy search: Speedy

- bounded-suboptimal search: EES
Contract Search
- A Direct Approach

- Vacillation
- DAS
- DAS Results

minimize cost subject to absolute time bound
minimize cost subject to absolute time bound
anytime algorithms: series of solutions, eg, unknown deadline
minimize cost subject to absolute time bound

anytime algorithms: series of solutions, eg, unknown deadline

while time remains
expand node with best reachable solution
minimize cost subject to absolute time bound
anytime algorithms: series of solutions, eg, unknown deadline

while time remains
expand node with best reachable solution

best: expected cost
reachable: expected search time to goal
how long to reach goal $\hat{d}(n)$ steps away?
how long to reach goal $\hat{d}(n)$ steps away?

how long to go one step toward a goal?

estimate $\Delta e$, time between a node’s generation and expansion
Vacillation

how long to reach goal $\hat{d}(n)$ steps away?

how long to go one step toward a goal?

estimate $\overline{\Delta e}$, time between a node’s generation and expansion

just record current time in node when generating

source #3 of 5: search vacillation

$$d_{max} = \text{time remaining} / \overline{\Delta e}$$
1. while time remains and open is not empty
2. \( d_{max} \leftarrow \) estimate bound
3. \( s \leftarrow \) pop best \( f \) from open
4. if \( s \) is a goal
5. save if best so far
6. else if \( \hat{d}(s) < d_{max} \)
7. expand \( s \)
8. else
9. add \( s \) to pruned
10. if open is empty, recover using pruned
DAS Results

Heuristic Search
Greedy Search
Bounded Search
Contract Search
  ■ Direct Approach
  ■ Vacillation
  ■ DAS
  □ DAS Results
Utility Functions
Conclusion

Dynamic Robot 500x500 (75 Lines)

Solution Quality

DAS
ARA* (wt=3.0)
ARA* (wt=1.5)
RWA* (wt=3.0)
CS

Deadline (seconds)

1/64 1/32 1/16 1/8 1/4 1/2 1 2 4 8 16 32 64
Search with a Utility Function
balance time and cost according to user’s utility function

Example: linear utility function

for solution of cost $f$ produced after time $t$:

$$U(f, t) = -w_f \cdot f - w_t \cdot t$$
balance time and cost according to user’s utility function

Example: linear utility function
for solution of cost $f$ produced after time $t$:

$$U(f, t) = -wf \cdot f - wt \cdot t$$

Example: minimize goal achievement time
if cost = plan makespan,

$$U(f, t) = -f - t$$
balance time and cost according to user’s utility function

Example: linear utility function
for solution of cost $f$ produced after time $t$:

$$U(f, t) = -w_f \cdot f - w_t \cdot t$$

Example: minimize goal achievement time
if cost = plan makespan,

$$U(f, t) = -f - t$$

anytime algorithms?
Requires a termination policy, assuming:

1. relevant solver features for predicting progress are known
2. training data available
3. new instance is similar in relevant aspects to training
4. relevant instance aspects are known
Requires a termination policy, assuming:
1. relevant solver features for predicting progress are known
2. training data available
3. new instance is similar in relevant aspects to training
4. relevant instance aspects are known

Impossible to design optimally:

\[ f = 5 \quad d = 1 \quad f = 4 \quad d = 2 \]

Must know the user’s trade-off!
source #4 of 5: user’s true objective/utility function

best-first search according to utility:

\[ U(n) = \max_{s \text{ under } n} (-w_f \cdot f(s) - w_t \cdot t(s)) \]
source #4 of 5: user’s true objective/utility function

best-first search according to utility:

\[ U(n) = \max_{s \text{ under } n} (-w_f \cdot f(s) - w_t \cdot t(s)) \]

convert \( \hat{d}(n) \) to \( t(n) \)

approximate \( s \) under \( n \) by cheapest and nearest

- need \( d_{\text{cheapest}}, d_{\text{nearest}}, h_{\text{cheapest}}, h_{\text{nearest}} \)
- source #5 of 5: cost-to-go to nearest, \( h_{\text{nearest}} \)
- seems straightforward in many domains
Different from anytime algorithms

- no need for termination policy (training data, precomputation)
- can spend all effort pursuing one solution
- no fixed trade-off

![Diagram showing cost, utility, and time with Bugsy and anytime solutions]
Different from anytime algorithms

- no need for termination policy (training data, precomputation)
- can spend all effort pursuing one solution
- no fixed trade-off
BUGSY Results

grid pathfinding

B — BUGSY
R — ARA* with termination policy learned off-line
A* — A*
S — Speedy

Wheeler Ruml (UNH)
Conclusion
Rational Search

Heuristic Search

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

Rational Search

Search

Summary

search algorithm

search strategy

open list

children, g, h values

domain

expand

Wheeler Ruml (UNH)
Rational Search

- Heuristic Search
- Greedy Search
- Bounded Search
- Contract Search
- Utility Functions
- Conclusion

Rational Search

Heuristic Search

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

Rational Search

Search

Summary

Heuristic Search

Greedy Search

Bounded Search

Contract Search

Utility Functions

Conclusion

Rational Search

Search

Summary

Wheeler Ruml (UNH)
Traditional:

1. initial state
2. goal predicate
3. expand — yields $g(n)$
4. $h(n)$
The Search Problem

Traditional:
1. initial state
2. goal predicate
3. expand — yields $g(n)$
4. $h(n)$

Possible:
5. $d(n)$: distance-to-go
6. $\hat{f}(n)$: expected cost, inadmissible $h$ (expansion experience)
7. $\Delta e$: expansion delay, time to goal (expansion experience)
8. $U(f, t)$: user’s utility function, true objective
9. $h_{\text{nearest}}(n)$: cost-to-go to nearest
search algorithms as agents
◆ expansion as sensor reading instead of proof step!
◆ connections:
  ■ reinforcement learning
  ■ metareasoning
  ■ decision-making

more information sources
 what else can we exploit? and how?

more problem settings
◆ greedy search: Speedy
◆ bounded-suboptimal search: EES
◆ contract search: DAS
◆ utility-based search: BUGSY
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