To Max or not to Max: Online Learning for Speeding Up Optimal Planning

C. Domshlak  E. Karpas  S. Markovitch

Faculty of Industrial Engineering and Management

Faculty of Computer Science
Technion

July 4, 2010
We want to do domain independent optimal planning, in a time-bounded setting

Use $A^*$
Motivation

- We want to do domain independent optimal planning, in a time-bounded setting
- Use $A^*$

$$f = g + h$$
We want to do domain independent optimal planning, in a time-bounded setting

Use $A^*$
Motivation

- We want to do domain independent optimal planning, in a time-bounded setting
- Use $A^*$

$$f = g + h$$

Which heuristic is the best?
Why Settle for One?

- There is no single best heuristic, so why settle only for one?
- We can use the maximum of several heuristics to get a more informative heuristic
Why Settle for One?

- There is no single best heuristic, so why settle only for one?
- We can use the maximum of several heuristics to get a more informative heuristic

Sample results:

<table>
<thead>
<tr>
<th>Domain</th>
<th>(h_l)</th>
<th>(h_{LM-CUT})</th>
<th>(h_{max})</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport</td>
<td>25</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>freecell</td>
<td>28</td>
<td>15</td>
<td>22</td>
</tr>
</tbody>
</table>

Number of problems solved in 30 minutes

A more informed heuristic solves less problems — something is rotten in the kingdom of \(A^*\).
Why Settle for One?

- There is no single best heuristic, so why settle only for one?
- We can use the maximum of several heuristics to get a more informative heuristic
- Sample results:

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h_{LA}$</th>
<th>$h_{LM-CUT}$</th>
<th>max$_h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport</td>
<td>25</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>freecell</td>
<td>28</td>
<td>15</td>
<td>22</td>
</tr>
</tbody>
</table>

Number of problems solved in 30 minutes
Why Settle for One?

- There is no single best heuristic, so why settle only for one?
- We can use the maximum of several heuristics to get a more informative heuristic

Sample results:

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h_{LA}$</th>
<th>$h_{LM-CUT}$</th>
<th>$\max_h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport</td>
<td>25</td>
<td><strong>38</strong></td>
<td>36</td>
</tr>
<tr>
<td>freecell</td>
<td><strong>28</strong></td>
<td>15</td>
<td>22</td>
</tr>
</tbody>
</table>

Number of problems solved in 30 minutes

- A more informed heuristic solves less problems — something is rotten in the kingdom of $A^*$
The Accuracy / Computation Time Tradeoff

- More Informed Heuristic
- Less Search Effort

Conclusion
A more informed heuristic is not necessarily better.
The Accuracy / Computation Time Tradeoff

More Informed Heuristic -> Less Search Effort -> Less Expanded States

Theoretical Model
From Model to Practice
Experimental Evaluation
The Accuracy / Computation Time Tradeoff

More Informed Heuristic

- Less Search Effort
- Less Expanded States
- More Time Per State

Conclusion

A more informed heuristic is not necessarily better
The Accuracy / Computation Time Tradeoff

Motivation

Theoretical Model

From Model to Practice

Experimental Evaluation

More Informed Heuristic

Less Search Effort

Less Expanded States

More Time Per State

\[ t_{maxh} = t_{h_{LA}} + t_{h_{LM-CUT}} \]
The Accuracy / Computation Time Tradeoff

Motivation

Theoretical Model

From Model to Practice

Experimental Evaluation

The Accuracy / Computation Time Tradeoff

More Informed Heuristic → Less Search Effort

Less Expanded States → More Time Per State

$t_{\text{max}_h} = t_{h_{\text{LA}}} + t_{h_{\text{LM-CUT}}}$

Conclusion

A more informed heuristic is not necessarily better
A Simple Observation

- So how can we benefit from multiple heuristics?

- Simple observation: the maximum of several heuristics — is simply the value of one of those heuristics

- This leads to the following idea:
  - Given state $s$, and heuristics $\{h_1, \ldots, h_n\}$
  - Choose $h_i = \text{ORACLE}(s, \{h_1, \ldots, h_n\})$
  - Compute only $h_i(s)$
A Simple Observation

So how can we benefit from multiple heuristics?

Simple observation: the maximum of several heuristics — is simply the value of one of those heuristics

This leads to the following idea:

- Given state $s$, and heuristics $\{h_1, \ldots, h_n\}$
- Choose $h_i = \text{ORACLE}(s, \{h_1, \ldots, h_n\})$
- Compute only $h_i(s)$
A Simple Observation

- So how can we benefit from multiple heuristics?

- Simple observation: the maximum of several heuristics — is simply the value of one of those heuristics

- This leads to the following idea:
  - Given state $s$, and heuristics $\{h_1, \ldots, h_n\}$
  - Choose $h_i = \text{ORACLE}(s, \{h_1, \ldots, h_n\})$
  - Compute only $h_i(s)$
The Oracle

- How do we define ORACLE?
  - Naive answer: use the heuristic which gives the maximum value
    \[
    \text{ORACLE}(s, \{h_1, \ldots, h_n\}) = \arg\max_i h_i(s)
    \]
  - Why is this naive?
    - Because sometimes the extra time to compute the most informed heuristic is not worth it
    - Example: \( h_{LM-CUT} \) is about 9.4 times slower than \( h_{LA} \)
How do we define ORACLE?

- Naive answer: use the heuristic which gives the maximum value

\[
\text{ORACLE}(s, \{h_1, \ldots, h_n\}) = \arg\max_i h_i(s)
\]

Why is this naive?

- Because sometimes the extra time to compute the most informed heuristic is not worth it
- Example: \( h_{\text{LM-CUT}} \) is about 9.4 times slower than \( h_{\text{LA}} \)
How do we define ORACLE?

Naive answer: use the heuristic which gives the maximum value

ORACLE(s, \{h_1, \ldots, h_n\}) = \arg\max_i h_i(s)

Why is this naive?

Because sometimes the extra time to compute the most informed heuristic is not worth it

Example: \(h_{LM-CUT}\) is about 9.4 times slower than \(h_{LA}\)
The Oracle

- How do we define ORACLE?
  - Naive answer: use the heuristic which gives the maximum value
    \[
    \text{ORACLE}(s, \{h_1, \ldots, h_n\}) = \arg\max_i h_i(s)
    \]
- Why is this naive?
  - Because sometimes the extra time to compute the most informed heuristic is not worth it
    - Example: \( h_{\text{LM-CUT}} \) is about 9.4 times slower than \( h_{\text{LA}} \)
The Oracle

- How do we define ORACLE?
  - Naive answer: use the heuristic which gives the maximum value
    \[
    \text{ORACLE}(s, \{h_1, \ldots, h_n\}) = \arg\max_i h_i(s)
    \]
  - Why is this naive?
  - Because sometimes the extra time to compute the most informed heuristic is not worth it
  - Example: $h_{\text{LM-CUT}}$ is about 9.4 times slower than $h_{\text{LA}}$
Our Contributions

- We develop a theoretical model for determining which heuristic is best to compute at each state, in order to minimize search time.
- We derive a decision rule from the model, which is used as a target concept for a classifier.
- We describe an online learning scheme which uses this classifier during search.
Outline

1. Motivation

2. Theoretical Model

3. From Model to Practice
   - Dealing with Model Assumptions
   - Learning
   - Using the Classifier

4. Experimental Evaluation
Assumptions

- State space is a tree
- Single goal state
- Uniform cost actions
- Constant branching factor $b$
- Perfect knowledge

Two heuristics: $h_1$ and $h_2$

- Consistent
- Evaluating $h_i$ takes time $t_i$
Theoretical Model - Which Heuristic to Compute When?

\[ f_1 = c^* \]
\[ f_2 = c^* \]

Surely expanded using \( h_1 \)

Best decision — expand, don't evaluate

Look at state \( s \) along the border

\[ f_2(s) = c^* \]

What is the best decision?

Using \( h_2 \) — 1 evaluation, \( t_2 \) time

Using \( h_1 \) — 1 evaluation, \( t_1 \) time

Best decision — use \( h_2 \) iff \( t_2 < b^\ell t_1 \)

\[ \ell > \log_b \left( \frac{t_2}{t_1} \right) \]

Estimating \( \ell \)

We make one more assumption: \( h_1 \) increases by \( c \) for each level.

Then \( \ell = h_2 - h_1 - 1 \).

Decision rule — use \( h_2 \) iff \( h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right) \)

\( \alpha \) is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

Surely expanded using $h_1$

$f_2 = c^*$

$f_1 = c^*$
Theoretical Model - Which Heuristic to Compute When?

Surely expanded using $\max_h$

$f_2 = c^*$

$f_1 = c^*$
Theoretical Model - Which Heuristic to Compute When?

Best decision — expand, don’t evaluate

Surely expanded using max\(_h\)

Best decision — use \(h_2\) iff \(t_2 < b^\ell t_1\)

Estimating \(\ell\)

We make one more assumption:

\(h_1\) increases by \(c\) for each level.

Then \(\ell = h_2 - h_1 + 1\).

Decision rule — use \(h_2\) iff

\[h_2 - h_1 > \alpha \log_b \left(\frac{t_2}{t_1}\right)\]

\(\alpha\) is a hyper-parameter.
Theoretical Model - Which Heuristic to Compute When?

Surely expanded using $h_1$.

Best decision — expand, don't evaluate

$f_1 = c^* f_2 = c^*$

Look at state $s$ along the border

We need to expand this region

And evaluate these states

$\ell$ — number of levels to expand

$b_\ell$ time

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$ — $b_\ell$ time

Best decision — use $h_2$ iff $t_2 < b_\ell t_1$

i.e. $\ell > \log b \left( \frac{t_2}{t_1} \right)$

Estimating $\ell$

We make one more assumption: $h_1$ increases by $c$ for each level.

Then $\ell = h_2 - h_1 + 1$.

Decision rule - use $h_2$ iff $h_2 - h_1 > \alpha \log b \left( \frac{t_2}{t_1} \right)$

$\alpha$ is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

- $f_2(s) = c^*$
- $f_1(s) < c^*$
Theoretical Model - Which Heuristic to Compute When?

What is the best decision?

\[ f_2(s) = c^* \]

\[ f_1(s) < c^* \]

\[ s_0 \]

\[ s_g \]

Decision rule - use \( h_2 \) iff \( h_2 - h_1 > \alpha \log_b (t_2 / t_1) \)

\[ \alpha \] is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time
Theoretical Model - Which Heuristic to Compute When?

Using $h_1$ — $1$ evaluation, $t_2$ time

Using $h_2$ — $1$ evaluation, $t_2$ time
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$

We need to expand this region

$S_0$

$S_g$

Decisions based on $h_2$ and $h_1$

Decision rule - use $h_2$ iff $h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right)$

$\alpha$ is a hyper-parameter

Estimating $\ell$

$\ell = h_2 - h_1 + 1$

We make one more assumption: $h_1$ increases by $c$ for each level.

Decision rule - use $h_2$ iff $h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right)$
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$

Expand region

And evaluate these states

Estimating $\ell$

We make one more assumption:

$h_1$ increases by $c$ for each level.

Then $\ell = h_2 - h_1 + 1$.

Decision rule - use $h_2$ iff $h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right)$

$\alpha$ is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$
Expanding region
Evaluating leaves

We need to expand this region
Expand region
Evaluate leaves

Estimating $\ell$
We make one more assumption:
$h_1$ increases by $c$ for each level.

Then $\ell = h_2 - h_1 c + 1$.

Decision rule — use $h_2$ iff $h_2 - h_1 > \alpha \log b (t_2 / t_1)$

$\alpha$ is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$

Expand region
Evaluate leaves

$b^\ell t_1$ time

$\ell$ — number of levels to expand

$s_0$

$s_g$

Using $h_2$ — 1 evaluation, $t_2$ time

Using $h_1$

Expand region
Evaluate leaves

$b^\ell t_1$ time

$\ell$ — number of levels to expand
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time
Using $h_1$ — $b^\ell t_1$ time

Using $h_2$ — 1 evaluation, $t_2$ time
Using $h_1$ — $b^\ell t_1$ time
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time
Using $h_1$ — $b^\ell t_1$ time

Best decision — use $h_2$ iff $t_2 < b^\ell t_1$

Using $h_2$ — 1 evaluation, $t_2$ time
Using $h_1$ — $b^\ell t_1$ time

Best decision — use $h_2$ iff $t_2 < b^\ell t_1$
Theoretical Model - Which Heuristic to Compute When?

Using $h_2$ — 1 evaluation, $t_2$ time
Using $h_1$ — $b^\ell t_1$ time

Best decision — use $h_2$ iff $t_2 < b^\ell t_1$

i.e. $\ell > \log_b \left( \frac{t_2}{t_1} \right)$
Theoretical Model - Which Heuristic to Compute When?

\[ \ell > \log_b \left( \frac{t_2}{t_1} \right) \]

Estimating \( \ell \)

- Expand region
- Evaluate leaves
- \( \ell \) — number of levels to expand
- \( b \ell \) — time

Best decision — use \( h_2 \) iff

\[ h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right) \]

\( \alpha \) is a hyper-parameter
Theoretical Model - Which Heuristic to Compute When?

\[ \ell > \log_b \left( \frac{t_2}{t_1} \right) \]

We make one more assumption: 
\( h_1 \) increases by \( c \) for each level.

Then \( \ell = \frac{h_2 - h_1}{c+1} \).
Theoretical Model - Which Heuristic to Compute When?

Decision rule - use $h_2$ iff

$$h_2 - h_1 > \alpha \log_b \left( \frac{t_2}{t_1} \right)$$

$\alpha$ is a hyper-parameter
Outline

1. Motivation
2. Theoretical Model
3. From Model to Practice
   - Dealing with Model Assumptions
   - Learning
   - Using the Classifier
4. Experimental Evaluation
Dealing with Model Assumptions

Assumptions

- State space is a tree
- Single goal state
- Uniform cost actions
- Constant branching factor $b$
- Perfect knowledge

Two heuristics: $h_1$ and $h_2$

- Consistent
- Evaluating $h_i$ takes time $t_i$
Dealing with Model Assumptions

Assumptions

- State space is a tree - rule is still applicable (possibly suboptimal)
- Single goal state - rule is still applicable (possibly suboptimal)
- Uniform cost actions - rule is still applicable (possibly suboptimal)
- Constant branching factor $b$
- Perfect knowledge

Two heuristics: $h_1$ and $h_2$

- Consistent - rule is still applicable (possibly suboptimal)
- Evaluating $h_i$ takes time $t_i$
Dealing with Model Assumptions

Assumptions

- State space is a tree - rule is still applicable (possibly suboptimal)
- Single goal state - rule is still applicable (possibly suboptimal)
- Uniform cost actions - rule is still applicable (possibly suboptimal)
- Constant branching factor $b$ - estimate
- Perfect knowledge

Two heuristics: $h_1$ and $h_2$

- Consistent - rule is still applicable (possibly suboptimal)
- Evaluating $h_i$ takes time $t_i$ - estimate
Dealing with Model Assumptions

Assumptions

- State space is a tree - rule is still applicable (possibly suboptimal)
- Single goal state - rule is still applicable (possibly suboptimal)
- Uniform cost actions - rule is still applicable (possibly suboptimal)
- Constant branching factor $b$ - estimate
- Perfect knowledge - use decision rule at every state

Two heuristics: $h_1$ and $h_2$

- Consistent - rule is still applicable (possibly suboptimal)
- Evaluating $h_i$ takes time $t_i$ - estimate
Learning

- **Pre-search:**
  - Collecting training examples
  - Labeling training examples
  - Generating features
  - Building a classifier

- **During search:**
  - Classification
  - Active learning
Collecting Training Examples

- State space is sampled using stochastic hill climbing “probes”
  - Depth limit $= 2 \times h(s_0)$
  - Probability of expanding successor $s \sim 1/h(s)$
- All *generated* states are added to the training set
- Probing stops when enough training examples are collected
Labeling Training Examples

- $b, t_1, t_2$ are estimated from the collected examples
- $h_2 - h_1$ is calculated for each state
- Each example is labeled by $h_2$ iff $h_2 - h_1 > \alpha \log_b(t_2/t_1)$

- WLOG $t_2 > t_1$ - the choice is always whether to evaluate the more expensive heuristic
Generating Features

- We perform online learning, for a specific problem, so we do not need to generalize across problems.
- This allows us to use features which fully describe each state.
- We use the simplest features - just values of state variables.
- Better features will probably lead to better results.
We use the Naive Bayes classifier

- Very fast
- Incremental — can be updated quickly on the fly
- Provides probability distribution for classification
Using the classifier

State Evaluation

\[
\text{state} \quad \text{features} \quad \text{classifier} \\
\downarrow \\
\text{Evaluate } h_1 \quad \text{Evaluate } h_2
\]

\[\Pr(h_1) > \rho \quad \Pr(h_2) > \rho \]

\[\Pr(h_1), \Pr(h_2) \leq \rho\]
Using the classifier

State Evaluation

- Pr($h_1$) > $\rho$ → Evaluate $h_1$
- Pr($h_2$) > $\rho$ → Evaluate $h_2$
- Pr($h_1$), Pr($h_2$) ≤ $\rho$ → Learn

Pr($h_1$) > $\rho$
Pr($h_2$) > $\rho$
Pr($h_1$), Pr($h_2$) ≤ $\rho$
Final Remarks

- This is an active online learning scheme

- Using multi-valued variable representation (and not STRIPS) helps, because it reduces dependence between state variables

- This approach can be easily extended to multiple heuristics
  - Learn a classifier for each pair
  - Decide which heuristic to use by voting
Outline

1. Motivation
2. Theoretical Model
3. From Model to Practice
   - Dealing with Model Assumptions
   - Learning
   - Using the Classifier
4. Experimental Evaluation
Evaluation

- We evaluated on problems from 22 domains from IPC 1 – 5
- We used two state of the art heuristics
  - $h_{LM-CUT}$ - Helmert and Domshlak 2009
  - $h_{LA}$ - Karpas and Domshlak 2009
- Parameters
  - $\alpha = 1$ - decision rule bias
  - $\rho = 0.6$ - confidence threshold
  - Training set size = 100
Anytime Behavior
Results - Time

Total Time to Solve All Common Problems

- $h_{LA}$
- $h_{LM-CUT}$
- $max_h$
- $rnd_h$
- $sel_h$
Conclusions

- It is possible to efficiently combine several admissible heuristics
- This leads to state-of-the-art performance

- Online learning can help in optimal planning

- I should probably read *Hamlet*
Conclusions

- It is possible to efficiently combine several admissible heuristics
- This leads to state-of-the-art performance

- Online learning can help in optimal planning

- I should probably read *Hamlet*
Thank You