

# 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

# Outline

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



# Motivation

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



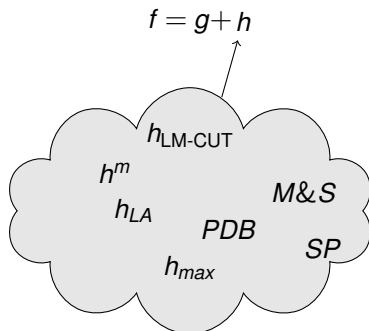
# Motivation

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

$$f = g + h$$

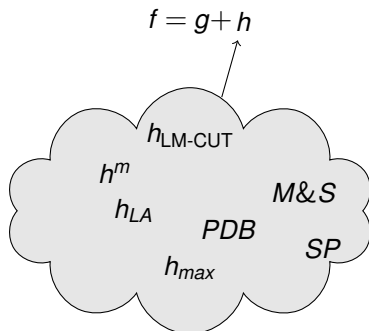
# Motivation

- 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  $A^*$



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



# 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:

Domain	$h_{LA}$	$h_{LM-CUT}$	$\max_h$
airport	25	<b>38</b>	36
freecell	<b>28</b>	15	22

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:

Domain	$h_{LA}$	$h_{LM-CUT}$	$\max_h$
airport	25	<b>38</b>	36
freecell	<b>28</b>	15	22

Number of problems solved in 30 minutes

- A more informed heuristic solves less problems — something is rotten in the kingdom of  $A^*$



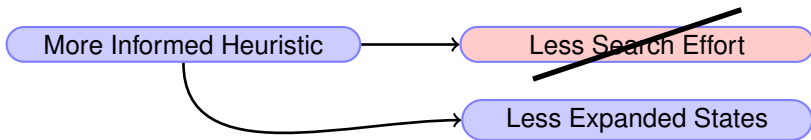
# The Accuracy / Computation Time Tradeoff

More Informed Heuristic

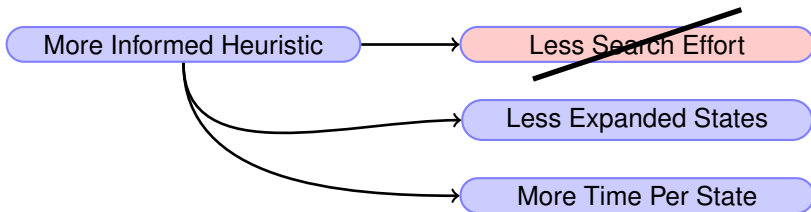


Less Search Effort

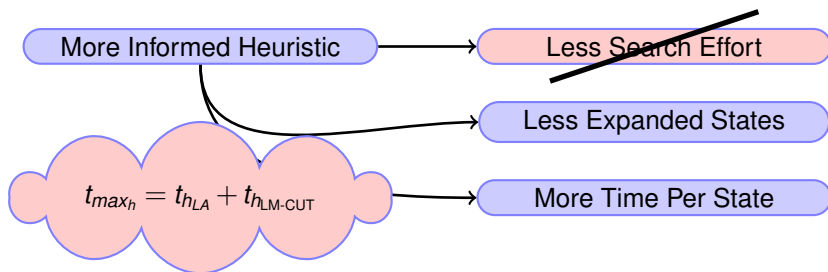
# The Accuracy / Computation Time Tradeoff



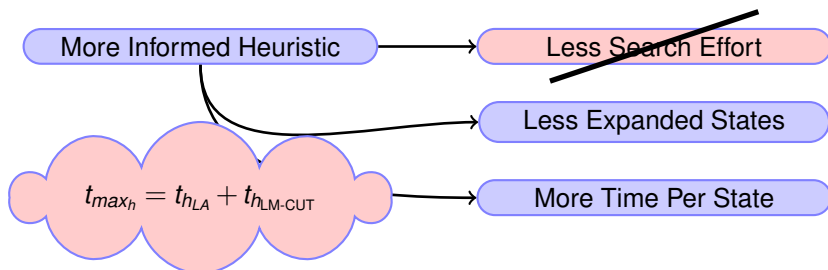
# The Accuracy / Computation Time Tradeoff



# The Accuracy / Computation Time Tradeoff



# The Accuracy / Computation Time Tradeoff



## 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  $\{h_1, \dots, h_n\}$
  - Choose  $h_i = \text{ORACLE}(s, \{h_1, \dots, h_n\})$
  - 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  $\{h_1, \dots, h_n\}$
  - Choose  $h_i = \text{ORACLE}(s, \{h_1, \dots, h_n\})$
  - 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  $\{h_1, \dots, h_n\}$
  - Choose  $h_i = \text{ORACLE}(s, \{h_1, \dots, h_n\})$
  - Compute only  $h_i(s)$



# The Oracle

- How do we define ORACLE?

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

$$\text{ORACLE}(s, \{h_1, \dots, h_n\}) = \underset{i}{\operatorname{argmax}} h_i(s)$$

- 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

$$\text{ORACLE}(s, \{h_1, \dots, h_n\}) = \underset{i}{\operatorname{argmax}} h_i(s)$$

- 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

$$\text{ORACLE}(s, \{h_1, \dots, h_n\}) = \underset{i}{\operatorname{argmax}} h_i(s)$$

- 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

$$\text{ORACLE}(s, \{h_1, \dots, h_n\}) = \underset{i}{\operatorname{argmax}} h_i(s)$$

- 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

$$\text{ORACLE}(s, \{h_1, \dots, h_n\}) = \underset{i}{\operatorname{argmax}} h_i(s)$$

- 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



# Theoretical Model - Which Heuristic to Compute When?

## Assumptions

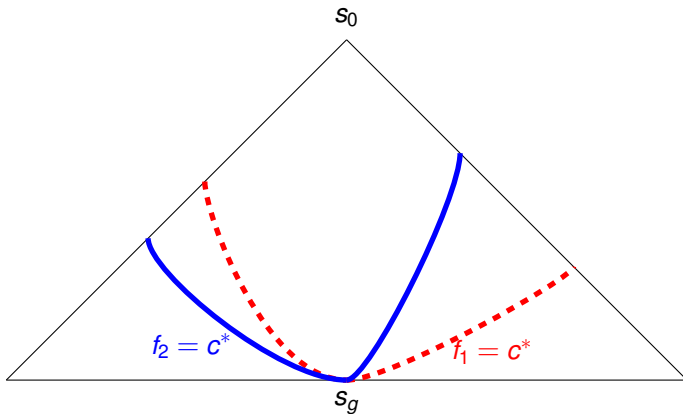
- State space is a tree
- Single goal state
- Uniform cost actions
- Constant branching factor  $b$
- Perfect knowledge

Two heuristics:  $h_1$  and  $h_2$

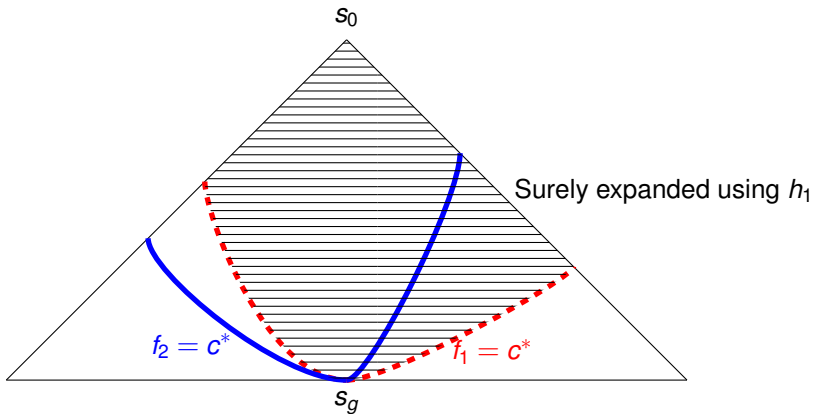
- Consistent
- Evaluating  $h_i$  takes time  $t_i$



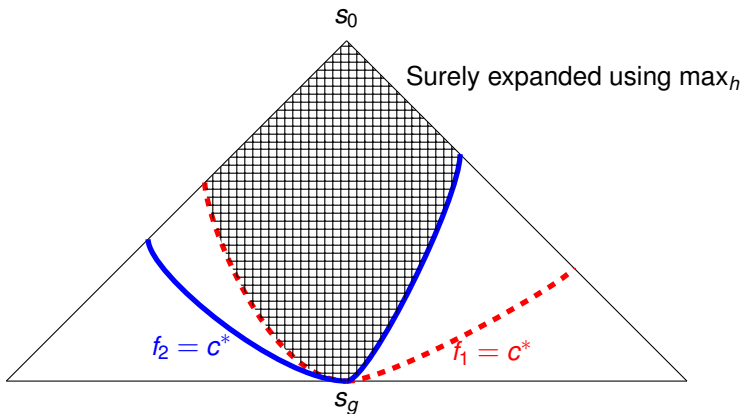
# Theoretical Model - Which Heuristic to Compute When?



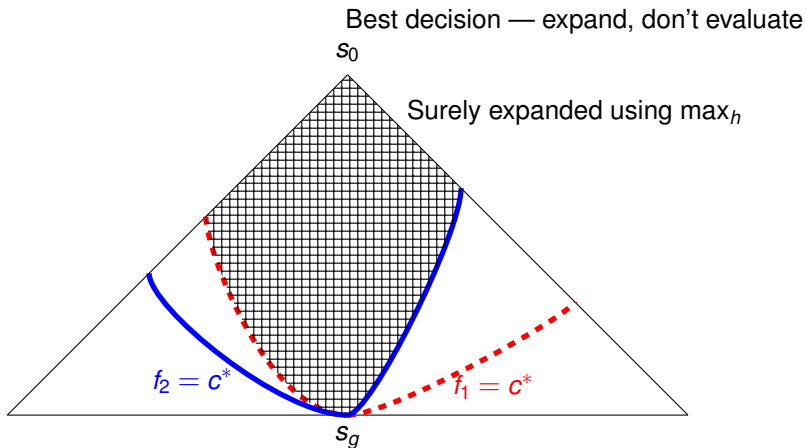
# Theoretical Model - Which Heuristic to Compute When?



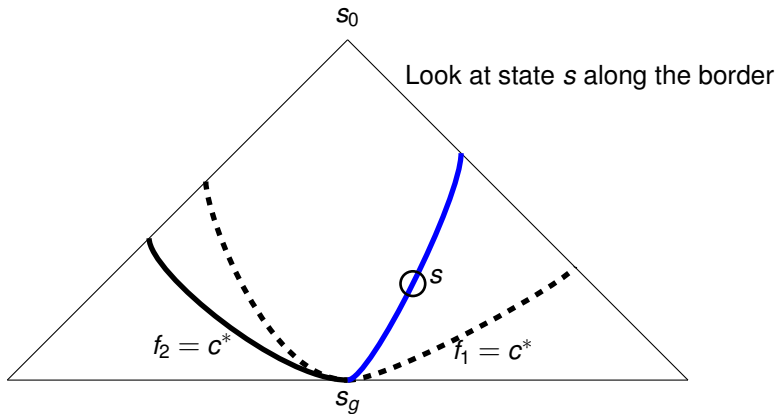
# Theoretical Model - Which Heuristic to Compute When?



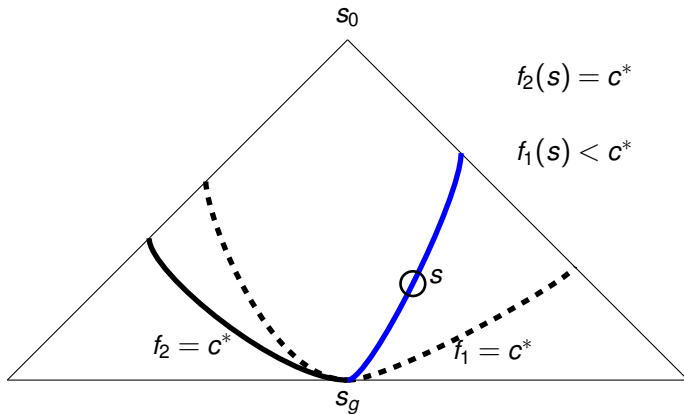
# Theoretical Model - Which Heuristic to Compute When?



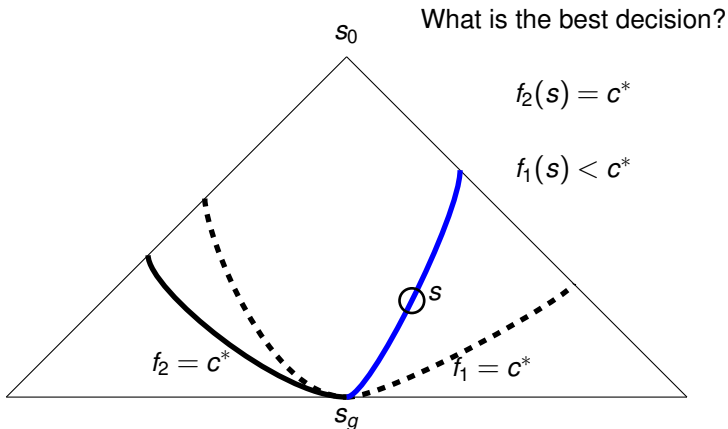
# Theoretical Model - Which Heuristic to Compute When?



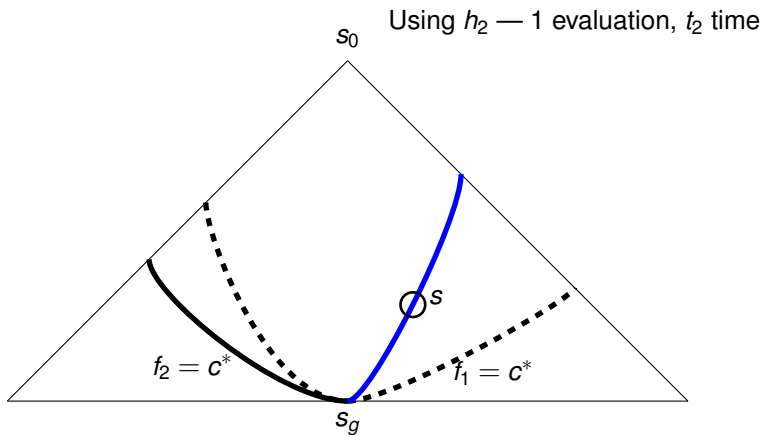
# Theoretical Model - Which Heuristic to Compute When?



# Theoretical Model - Which Heuristic to Compute When?



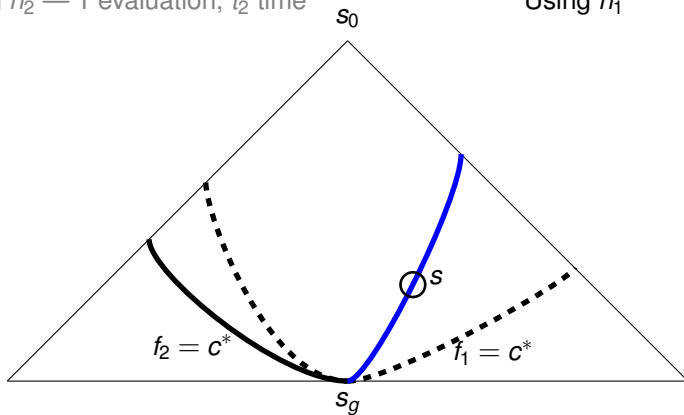
# Theoretical Model - Which Heuristic to Compute When?



# Theoretical Model - Which Heuristic to Compute When?

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

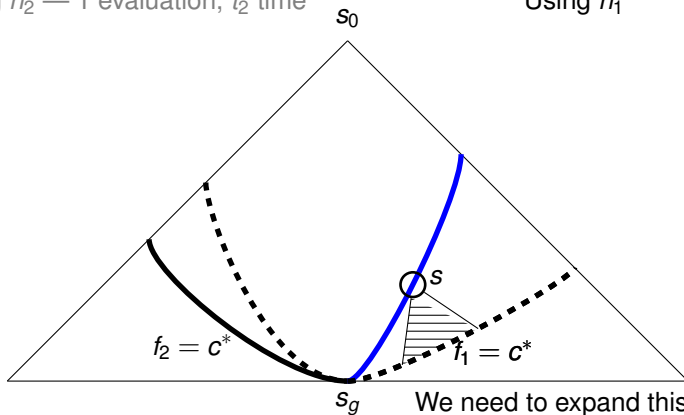
Using  $h_1$



# Theoretical Model - Which Heuristic to Compute When?

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

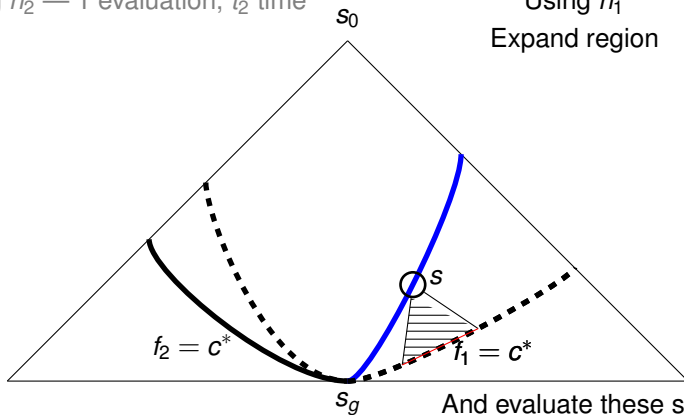
Using  $h_1$



# Theoretical Model - Which Heuristic to Compute When?

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

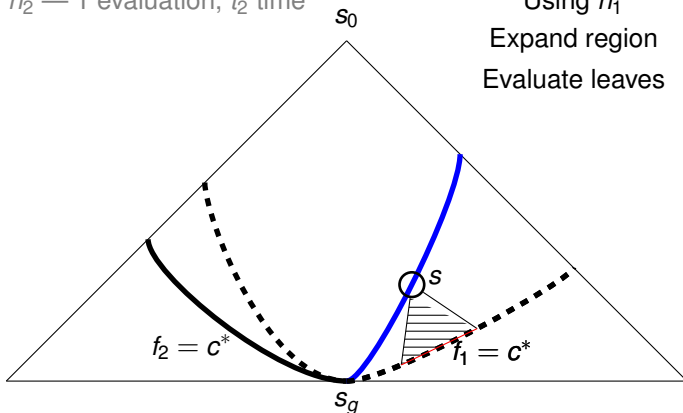
Using  $h_1$   
Expand region



# Theoretical Model - Which Heuristic to Compute When?

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

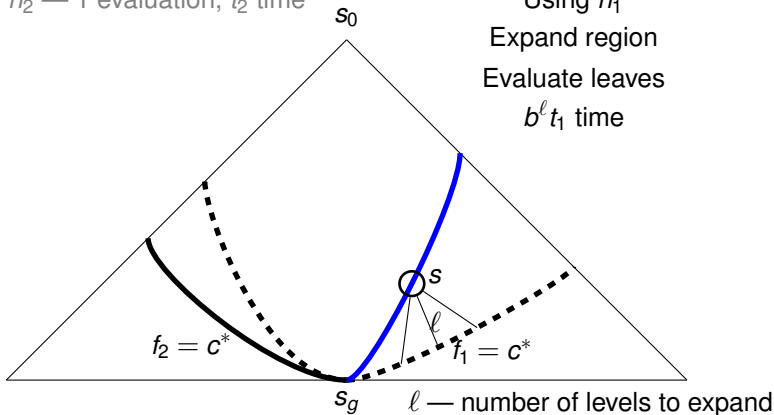
Using  $h_1$   
Expand region  
Evaluate leaves



# Theoretical Model - Which Heuristic to Compute When?

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

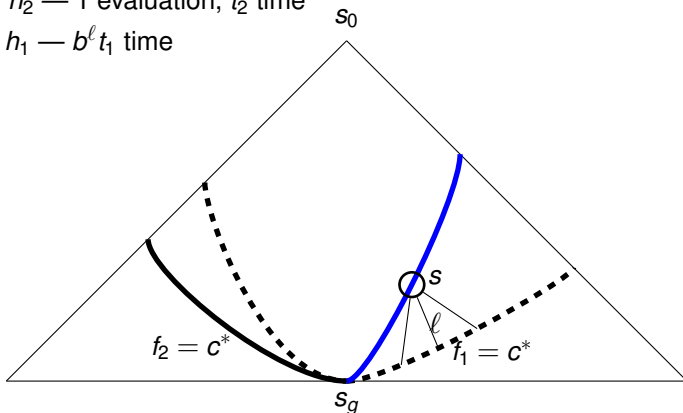
Using  $h_1$   
Expand region  
Evaluate leaves  
 $b^l t_1$  time



# Theoretical Model - Which Heuristic to Compute When?

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

Using  $h_1$  —  $b^l t_1$  time

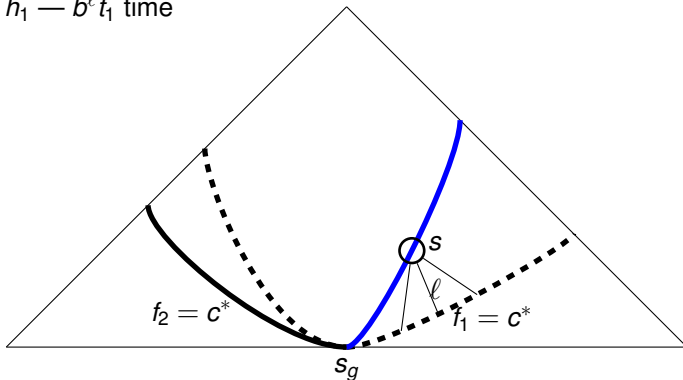


# Theoretical Model - Which Heuristic to Compute When?

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

Using  $h_1$  —  $b^l t_1$  time

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



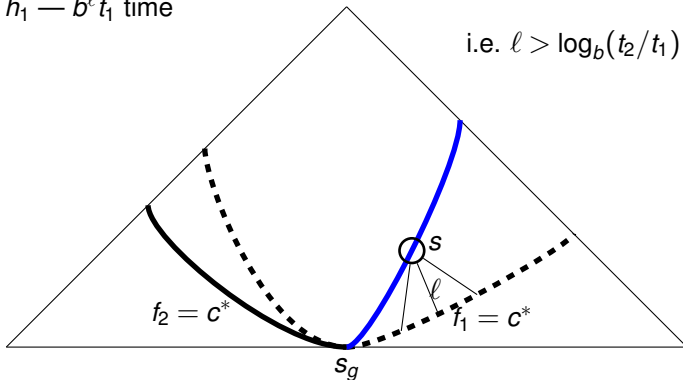
# Theoretical Model - Which Heuristic to Compute When?

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

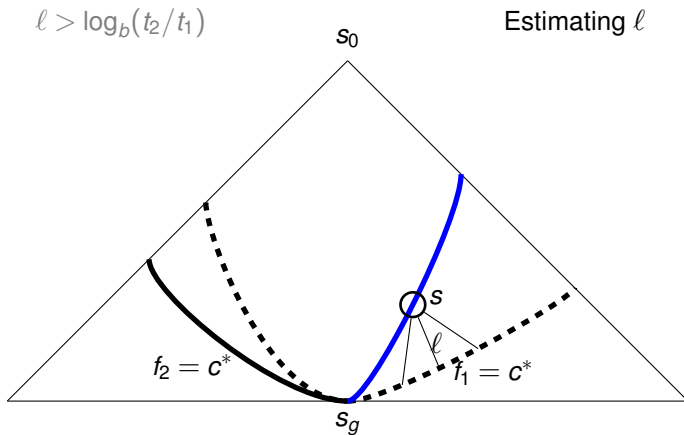
Using  $h_1$  —  $b^l t_1$  time

$s_0$  Best decision — use  $h_2$  iff  $t_2 < b^l t_1$

i.e.  $l > \log_b(t_2/t_1)$

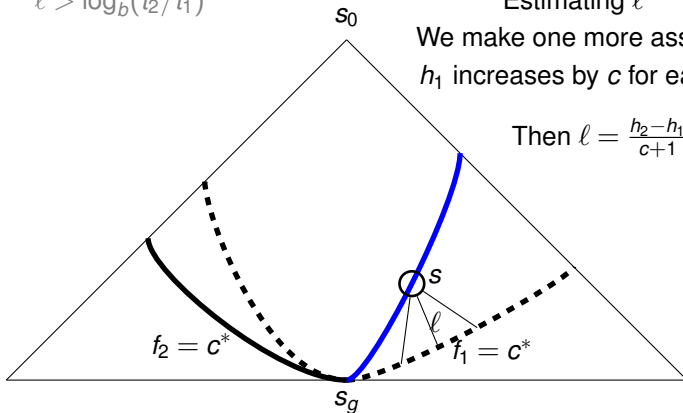


# Theoretical Model - Which Heuristic to Compute When?

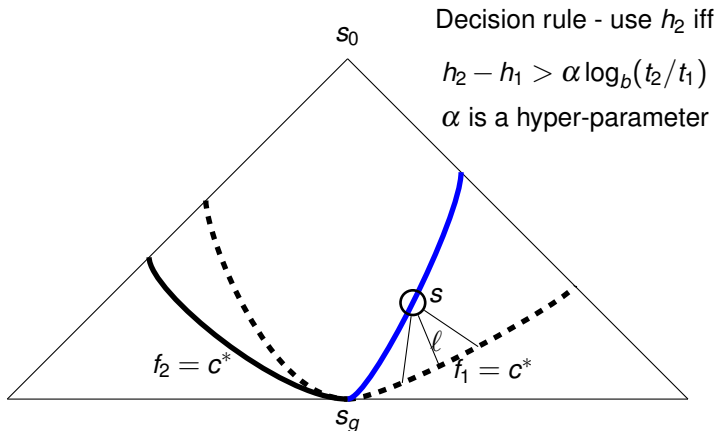


# Theoretical Model - Which Heuristic to Compute When?

$$l > \log_b(t_2/t_1)$$



# Theoretical Model - Which Heuristic to Compute When?



# 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

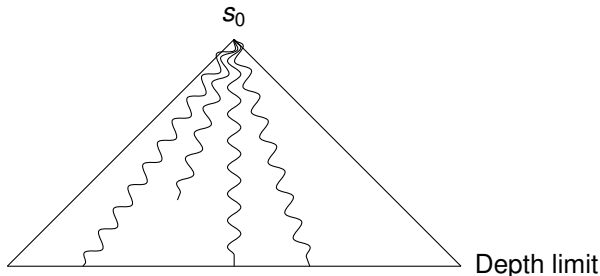


# Learning

- 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 =  $2 * h(s_0)$
  - 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



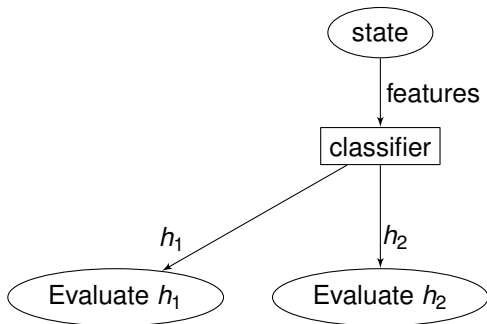
# Building a Classifier

- We use the Naive Bayes classifier
  - Very fast
  - Incremental — can be updated quickly on the fly
  - Provides probability distribution for classification



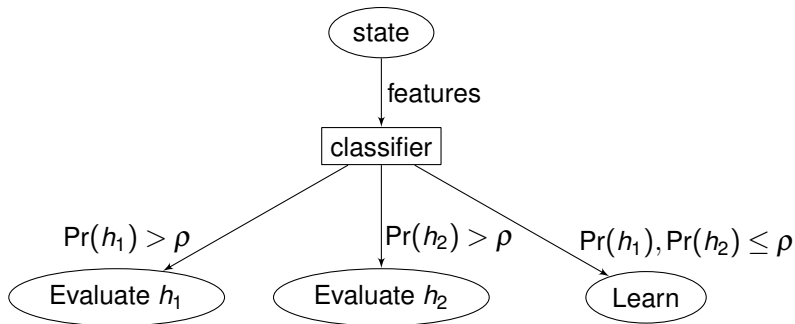
# Using the classifier

## State Evaluation



# Using the classifier

## State Evaluation



# 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

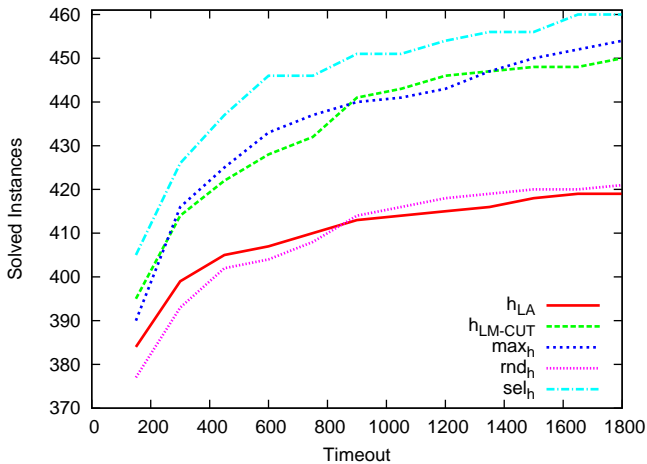


# Evaluation

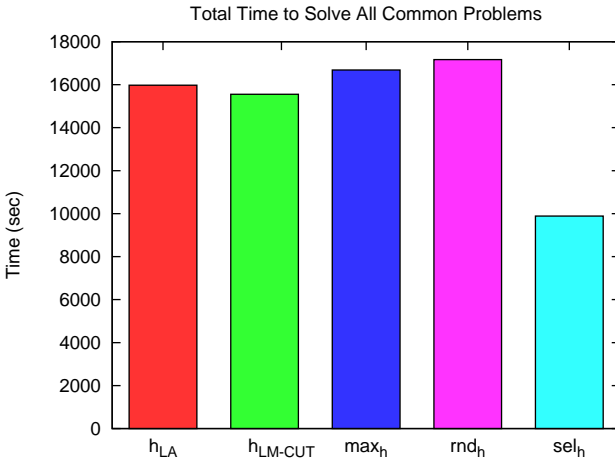
- We evaluated on problems from 22 domains from IPC 1 – 5
- We used two state of the art heuristics
  - $h_{\text{LM-CUT}}$  - Helmert and Domshlak 2009
  - $h_{\text{LA}}$  - Karpas and Domshlak 2009
- Parameters
  - $\alpha = 1$  - decision rule bias
  - $\rho = 0.6$  - confidence threshold
  - Training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



# Anytime Behavior



# Results - Time



# 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

- Thank You

