Learning to Combine Admissible Heuristics Under Bounded Time

Carmel Domshlak, Erez Karpas, Shaul Markovitch - Technion

Motivation

- Optimal planning $\equiv A^*$ + admissible heuristic (almost always)
- Which heuristic to use?
- Why use only one heuristic?
- Simplest combination method: $\max$

Domain	h_{LA}	$h_{\text{LM-CUT}}$	max
airport	25	**38**	36
freecell	**28**	15	22

The Problem with $\max$

- We need to compute many heuristic functions
- The heuristic value is the result of only one computation
- Some computation is wasted

Possible solution: learn a classifier which predicts which heuristic will be the "winner".

Caveat: sometimes spending a lot of time to compute the most informed heuristic is not the best thing to do (see the results from the sequential optimal track in IPC-2008).

Theoretical Model

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

Decision Rule

Above both lines is the surely expanded region. Best decision - just expand, don't evaluate.

For state s on the border, the options are either to use h_2 which takes time t_2, or use h_1, in which case we need to expanded the highlighted region, which take $b^l t_1$ time.

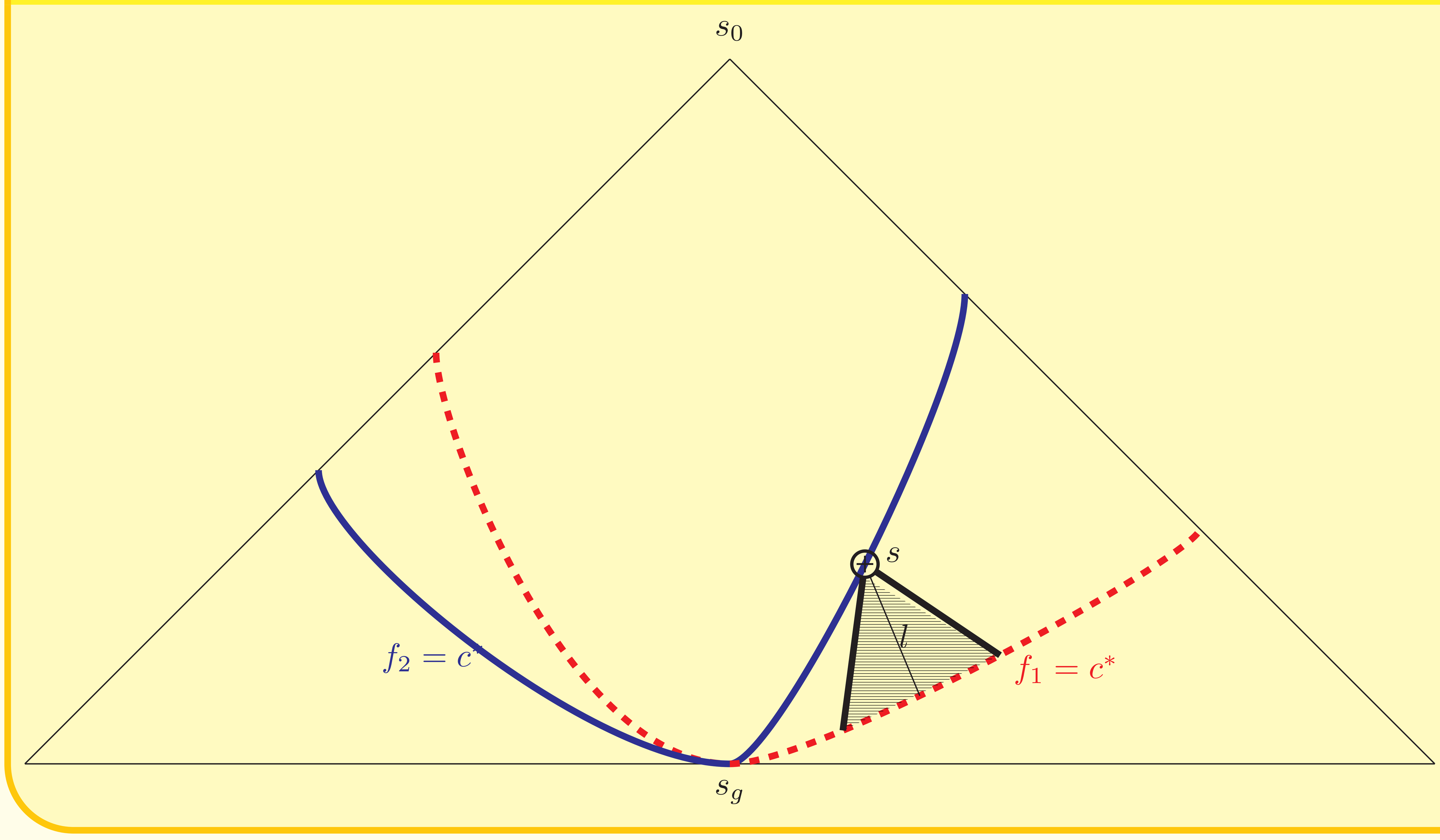
Therefore the best decision for s is to use h_2 iff $t_2 < b^l t_1$, or if $l > \log_b(t_2/t_1)$.

After some more assumptions about the rate of growth of heuristic error, we can write the rule as

$$\text{Use } h_2 \quad \text{iff} \quad h_2 - h_1 > \alpha \log_b(t_2/t_1)$$

α is a hyper-parameter

Theoretical Model Illustrated



Dealing with Assumptions

Assumptions

- State space is a tree - doesn't change the rule
- Single goal state - doesn't change the rule
- Uniform cost actions - doesn't change the rule
- Constant branching factor b - estimate
- Perfect knowledge - use decision rule at every state

Two heuristics: h_1 and h_2

- Consistent - doesn't change the rule
- Evaluating h_i takes time t_i - estimate

Labelling Examples

First b, t_1, t_2 are estimated from the collected examples. Then $h_2 - h_1$ 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.

Experiments

We used two state of the art heuristics: $h_{\text{LM-CUT}}$ (Helmert and Domshlak 2009) and h_{LA} (Karpas and Domshlak 2009).

Parameters were set to $\alpha = 1, \rho = 0.6$, Training set size = 100.

We compare to each of the individual heuristics, regular $\max$, and rnd_h, which selects one of the heuristics at random.

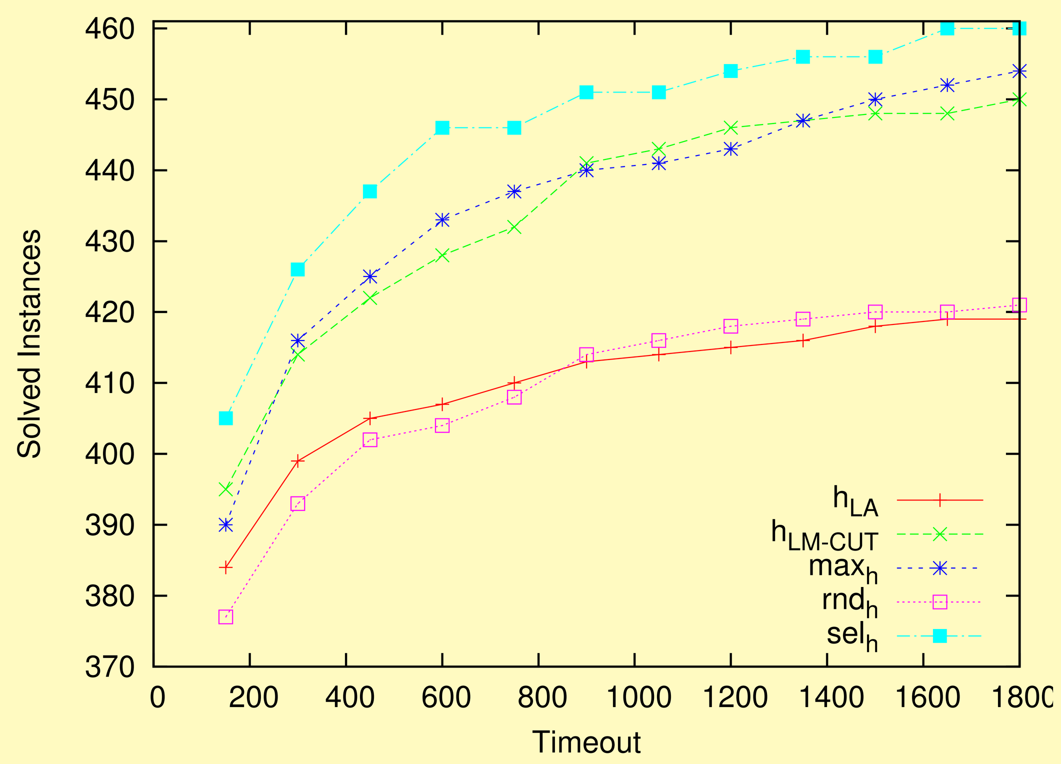
Learning

- Collecting training examples
- Labeling training examples
- Generating features
- Building a classifier

Features

We use the simplest features - values of state variables. Better features will probably lead to better results.

Experiments - Anytime Behavior



Classifier

We use the Naive Bayes classifier, because it is:

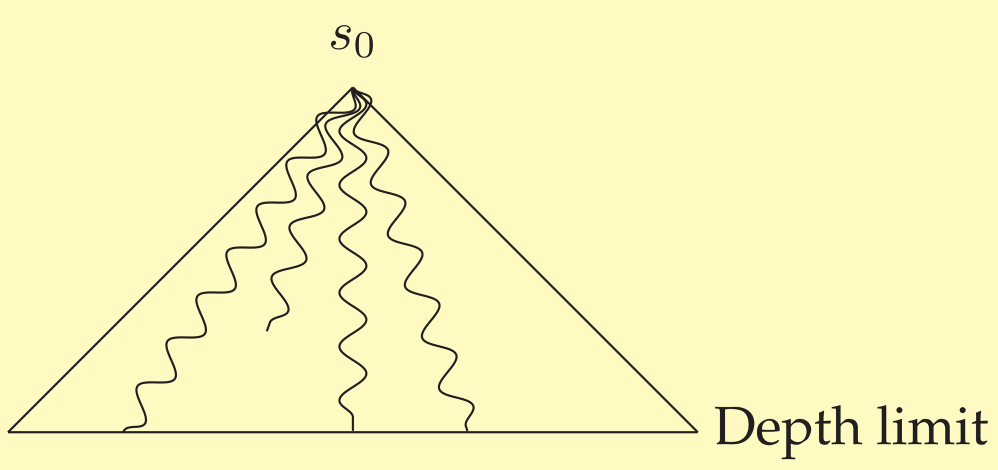
- Very fast
- Incremental
- Provides probabilistic classification

Collecting Examples

State space is sampled using stochastic hill climbing "probes"

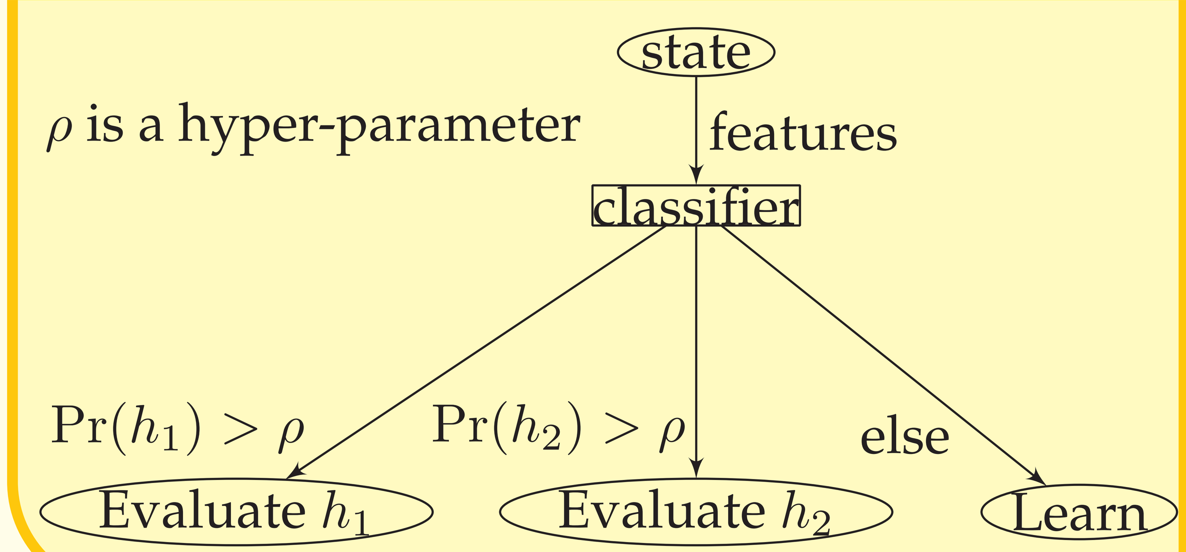
- Depth limit = $2 * h(s_0)$
- Probabiity of expanding successor s is $1/h(s)$

All *generated* states are added to the training set. Probing stops when enough training examples are collected



Using the Learned Model

ρ is a hyper-parameter



Experiments - Result Summary

	h_{LA}	$h_{\text{LM-CUT}}$	max	rnd_h	sel_h
Solved	419	450	454	421	**460**
Avg Time	39.7	38.6	41.4	42.6	**24.5**
Avg Expanded	35.4	8.2	**1**	4	1.6