When Optimal is Just Not Good Enough: Learning Fast Informative Action Cost Partitionings

Erez Karpas Michael Katz Shaul Markovitch

Faculty of Industrial Engineering and Management

Faculty of Computer Science
Technion

May 31, 2011

Outline

1. Motivation
2. General Framework
3. Algorithmic Details
4. Empirical Evaluation
   - Empirical Evaluation of Our Basic Assumption
   - Empirical Evaluation of Our Approach
5. Conclusion

Cost-Partitioning Based Heuristics

- Many state-of-the-art heuristics are based upon some form of action cost partitioning

Action Cost Partitioning

1. Divide the cost of each action between several subproblems (implicit abstractions, landmarks, . . .)
2. Obtain a heuristic estimate for each subproblem
3. The sum of estimates is admissible if each action contributes no more than its total cost

- A cost partitioning is optimal (for some state) if it yields the maximal heuristic estimate possible for that state

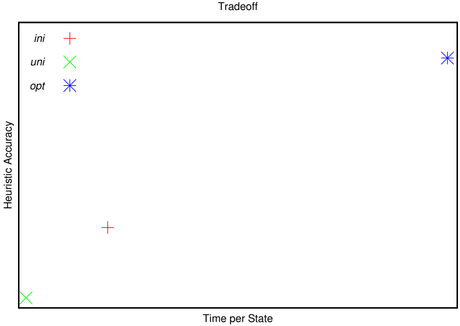
Motivation

- We focus on heuristics for which a polytime procedure for finding an optimal cost partitioning is known
- In all known cases so far, the procedure for finding an optimal cost partitioning involves solving a Linear Programming (LP) problem

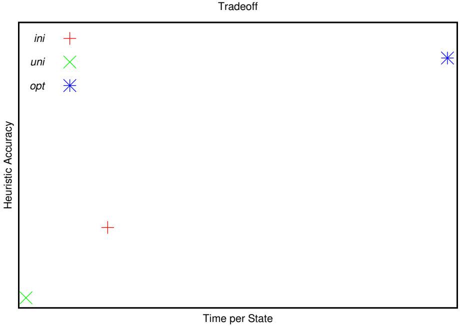
Cost Partitioning Schemes in Practice

- Optimal
  - SLOOOOOOOOOOOOOOW
  - Very informative
- Ad-hoc (usually uniform)
  - Very fast
  - Less informative
- A compromise: initial-optimal cost partitioning
  - Fast
  - Less informative

Time/Accuracy Tradeoff



Time/Accuracy Tradeoff



Goal: create a cost-partitioning based heuristic that allows control over its location in this tradeoff

Outline

Basic Assumption

- Our approach is based on the following assumption:

Assumption

An optimal cost partitioning for state s will give a “good” heuristic estimate for state s' if s and s' are “close”

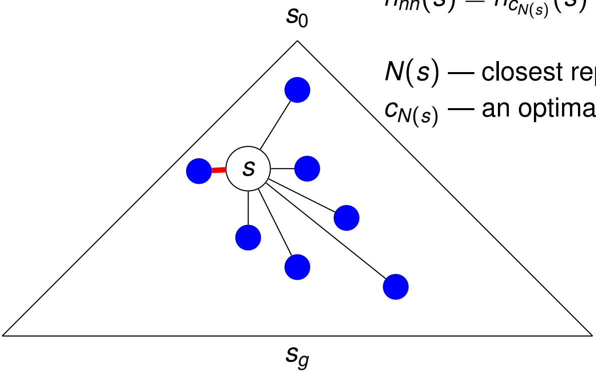
- We will formulate this assumption mathematically later, and provide an empirical evaluation that supports it
- “Close" is defined in terms of some metric between states d

Basic Framework

- Given a planning task, choose k states in a principled way
- Compute an optimal cost partitioning for each of these states
- During search, use the optimal cost partitionings of these k states to create a heuristic estimate

- Increasing k increases accuracy (at the cost of computation-time)

Heuristic Option 1: Nearest Neighbor



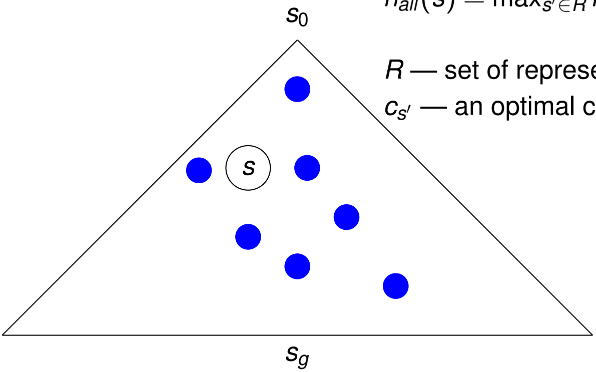
Use closest representative to evaluate s

$h_{nn}(s) = h_{c_{N(s)}}(s)$

$N(s)$ — closest representative to s

$c_{N(s)}$ — an optimal cost partitioning for $N(s)$

Heuristic Option 2: All k Representatives



or

Use all representatives to evaluate s

$h_{all}(s) = \max_{s' \in R} h_{c_{s'}}(s)$

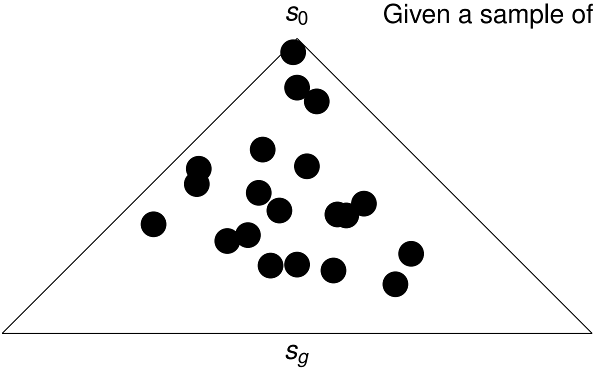
R — set of representative states

$c_{s'}$ — an optimal cost partitioning for s'

Choosing Representatives

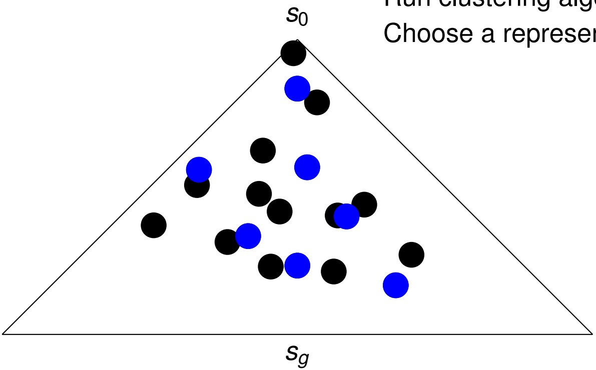
- How can we choose representatives in a principled way?
- We want to minimize the distance (according to the metric) from each state in the state space to the closest representative
- We can’t deal with the entire state space, so we use a sample

Choosing Representatives — Illustrated



Given a sample of states

Choosing Representatives — Illustrated



Run clustering algorithm

Choose a representative from each cluster

Outline

Filling in the Details

The framework above needs some details

- How to sample the state space?
- Which clustering algorithm to use?
- Which metric to use?

State Space Sample

- We use the sampling procedure of Haslum et. al. (2007)

Repeat 1000k times:

1. Choose depth $\mathscr{L}$ distributed binomially around the estimated goal depth
2. Perform a random walk up to depth $\mathscr{L}$ from initial state
3. Add last state in walk to sample

Clustering Algorithm

Requirements:

- We need to control the number of clusters k
- We need to get a representative for each cluster

Options:

- k-means seems like a good option, but what is the centroid of $on(A, B)$ and $on(A, C)$?
- We use k-medoids (Hartigan and Wong, 1979), which returns a median representative for each cluster

Metric (In Theory)

- Theoretically, we want to use the distance from s_1 to s_2 in the state space
- This is somewhat justified by abstraction based heuristics being consistent
- However:
  - The true distance is not symmetric, and might be infinite
  - The true distance is P-SPACE Complete to compute

Metric (In Practice) — d_s

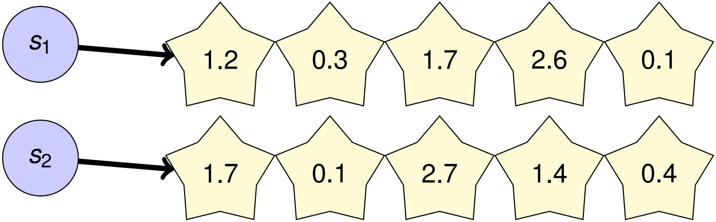
Number of Mismatching Variables:
$d_s(s_1, s_2) := |\{v \in V | s_1[v] \neq s_2[v]\}|$

s_1	1	0	1	1	0
s_2	1	1	0	1	0

$d_s(s_1, s_2) = 2$

Metric (In Practice) — d_e

The Euclidean distance between the vectors of estimate values of each subproblem under uniform cost partitioning: $d_e(s_1, s_2)$



$$d_e(s_1, s_2) = |\langle 1.2, 0.3, 1.7, 2.6, 0.1\rangle - \langle 1.7, 0.1, 2.7, 1.4, 0.4\rangle|_2$$

Outline

Empirical Evaluation

- We implemented our approach on top of Fast Downward, and evaluate it on implicit abstractions heuristics
- Initial results with h_{nn} were not promising, so we only evaluate h_{all}
- We evaluate clustering with d_s (clstr-s), clustering with d_e (clstr-e) as well as random choice of representatives (rand)
- We compare to optimal cost partitioning (opt), uniform cost partitioning (uni), and initial optimal cost partitioning (ini)
- With fork implicit abstractions, the resulting LP for optimal cost partitioning is too large to solve for many problems. Here, we present results only for inverted forks

Outline

Evaluating the Basic Assumption

To evaluate our basic assumption empirically, we must first formulate it in statistical terms:

Basic Assumption - Statistically Speaking

Let s, s' be two states, such that the minimal distance from s to s' is d. Denote the relative loss of accuracy from using an optimal cost partitioning of s' to evaluate s by

$$\Delta_{s,s'} := \frac{h_{C_s}(s) - h_{C_{s'}}(s)}{h_{C_s}(s)}.$$

Then $\Delta_{s,s'}$ and $\Delta_{s',s}$ are positively correlated with d.

Statistical Test

- We perform a statistical test of our hypothesis for each planning task
- We first obtain a sample of pairs of states, with known minimal distance by repeating the following 10 times:
  1. Sample a random state s using random walk
  2. Perform BFS from s, up to depth $h_{FF}(s_0)$
  3. From each layer l in the BFS, choose state s_l randomly
  4. Add Δ_{s,s_l} and $\Delta_{s_l,s}$ to sample, with minimal distance l
- Perform Kendall τ-b rank-correlation test on sample (ignoring tasks with less than 30 pairs in the sample)

Statistical Test - Results

Domain	Inverted Forks	
	Total	Significant ($p < 0.05$)
airport-ipc4	9	5
blocks-ipc2	19	16
depots-ipc3	0	0
driverlog-ipc3	6	6
freecell-ipc3	5	5
grid-ipc1	1	0
gripper-ipc1	7	7
logistics-ipc1	2	2
logistics-ipc2	10	10
miconic-strips-ipc2	43	43
mprime-ipc1	16	13
mystery-ipc1	18	12
openstacks-ipc5	0	0
pathways-ipc5	2	1
pw-notank-ipc4	15	13
pw-tank-ipc4	0	0
psr-small-ipc4	33	30
rovers-ipc5	7	7
satellite-ipc4	4	4
schedule-ipc2	41	25
tpp-ipc5	4	3
zenotravel-ipc3	7	7
TOTAL	249	209

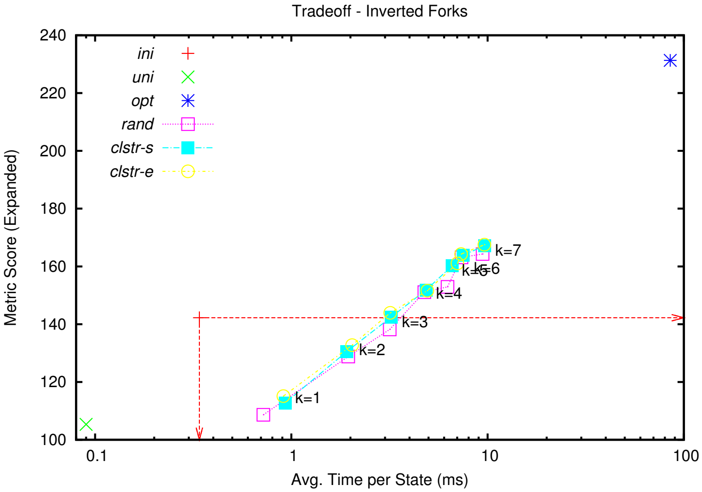
Overall: accept with $p < 0.05$ in 83.9% of planning tasks

Outline

Accuracy/Computation-Time Tradeoff

- We illustrate the tradeoff between heuristic accuracy and heuristic computation time, by plotting them together
- We show average heuristic computation-time per state on the x-axis (in logscale)
- We show informativeness on the y-axis, as measured by e_i/e^* where e_i is the number of states expanded by method i and e^* is the minimum over all e_i's
- Averages are over tasks solved by all methods

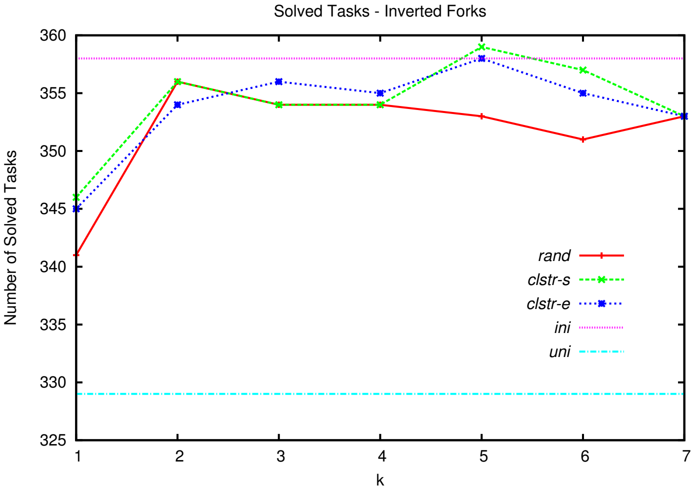
Accuracy/Computation-Time Tradeoff



Solved Tasks

- What really matters is the number of solved tasks
- We plot the number of solved tasks against k

Solved Tasks - Inverted Forks



Choosing the Best k

- No one is forcing us to use the same k everywhere
- If we choose the best k for each domain, we get much better results

Method	*ini*	*uni*	*opt*	*rand*	*clstr-s*	*clstr-e*
Inverted Forks	358	329	238	365	**369**	**369**

Outline

Conclusion

- We presented a method for fast, informative action cost partitioning
- This method allows us control over the computation-time/heuristic accuracy tradeoff
- The new cost partitioning can lead to solving more planning tasks

Thank You

- Thank You