

Predicting Polarity with Supervised Learning; Challenging SATzilla with the Borg Algorithm Portfolio; Visualizing Solver Performance at the 2011 SAT Competition

Bryan Silverthorn and Risto Miikkulainen
The University of Texas at Austin
{bsilvert,risto}@cs.utexas.edu

Learning to predict variable polarity

Most solvers treat each instance of a computational problem as independent. Different instances, however, are often related. When solving a new instance, can we reach a solution more quickly by leveraging knowledge gained from past instances? This work [2] explores the possibility of transferring successful *low-level* solver decisions from solved to unsolved instances, using fine-grained problem structure to identify areas of similarity.

Outline of the learning procedure

- Given a set of related satisfiable problem instances, some previously solved,
1. compute low-level structural features for every variable in solved instances,
 2. label each feature vector with the observed satisfying values of its variable,
 3. train an efficient binary classifier on these feature vectors, and
 4. use the classifier to intelligently initialize your solver for each new instance.

Experimental results: classification accuracy (logistic regression)

Feature	Misclassification Rate (%)							
	b10	qcp	gcp	cbmc	bms	ii	rti	
clause_length_mu	50	17	16	40	49	30	50	
clause_length_sigma	50	17	16	40	49	32	50	
clause_polarity_mu	40	17	16	40	49	25	40	
clause_polarity_sigma	50	17	16	40	49	25	50	
clauses	50	17	16	40	49	29	50	
consequents	44	17	16	43	50	24	44	
horn_clauses	41	17	16	40	46	29	42	
neighbors	50	17	16	40	49	33	50	
polarity	33	17	16	43	42	24	33	
nh_clause_polarity_mu_mu	45	17	16	40	51	48	45	
nh_clause_polarity_mu_sigma	50	17	16	29	49	43	50	
...								
nh_clauses_mu	50	17	16	37	49	47	50	
nh_clauses_sigma	50	17	16	40	49	47	50	
nh_consequents_mu	49	17	16	40	50	37	49	
nh_consequents_sigma	50	17	16	40	49	43	50	
nh_horn_clauses_mu	46	17	16	40	48	44	46	
nh_horn_clauses_sigma	50	17	16	30	49	41	50	
nh_neighbors_mu	50	17	16	40	49	46	50	
nh_neighbors_sigma	50	17	16	34	49	44	50	
nh_polarity_mu	46	17	16	40	51	38	46	
nh_polarity_sigma	50	17	16	31	49	43	50	
none	50	17	16	40	49	48	50	
all	33	17	16	22	41	20	33	

Experimental results: cost ratio vs. uniform (MiniSat and TNM)

Collection	decisions	(σ)	props	(σ)	steps	(σ)
SAPS-newQCP	0.99	0.07	1.00	0.08	1.00	0.03
SAPS-SWGCP	0.98	0.04	0.98	0.04	0.70	0.06
satenstein-cbmc	1.04	0.10	0.88	0.08	0.59	0.04
satlib-bms	0.97	0.05	0.97	0.06	1.01	0.23
satlib-cbs-m403-b10	0.76	0.03	0.72	0.03	0.79	0.04
satlib-cbs-m449-b90	0.93	0.02	0.92	0.03	0.92	0.08
satlib-ii	1.33	0.97	1.52	1.96	1.26	0.44
satlib-rti	0.87	0.04	0.85	0.04	0.87	0.13

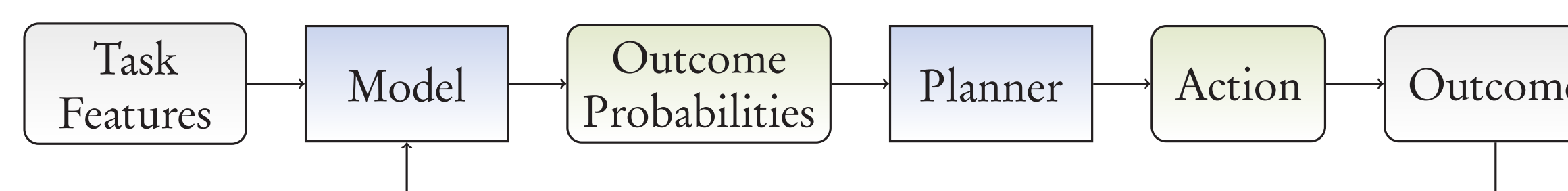
In most cases, classifier-based solver initialization measurably reduces search cost.

Borg: a multi-domain algorithm portfolio

Algorithm portfolios, such as SATzilla [3], automatically split CPU time among solvers, tailoring their allocations to individual problem instances. They are a promising approach to taming the growing “solver zoo”. Borg is a new algorithm portfolio architecture, focusing on moving the research area forward by

1. constructing more expressive probabilistic models of solver performance,
 2. considering a more general class of potential solver execution schedules, and
 3. developing a single high-quality implementation for multiple problem domains.
- Borg placed first at the 2010 pseudo-Boolean solver competition (PBS track), and more recent versions are competing in this year’s SAT and PB competitions.

Repeated model-based action selection



- The borg portfolio architecture repeatedly incorporates new evidence, replanning at each of a series of decision points. Until the instance is solved,
- ▶ the model computes a set of weighted alternative predictions, each of which includes a probability of success for every solver and every discrete timeout;
 - ▶ the planner computes a complete schedule for the time remaining, attempting to maximize the overall predicted probability that some solver will succeed;
 - ▶ a solver run is initiated or unpaused based on the planned schedule, allowed to execute for some amount of CPU time, then paused; and
 - ▶ the decision process repeats.

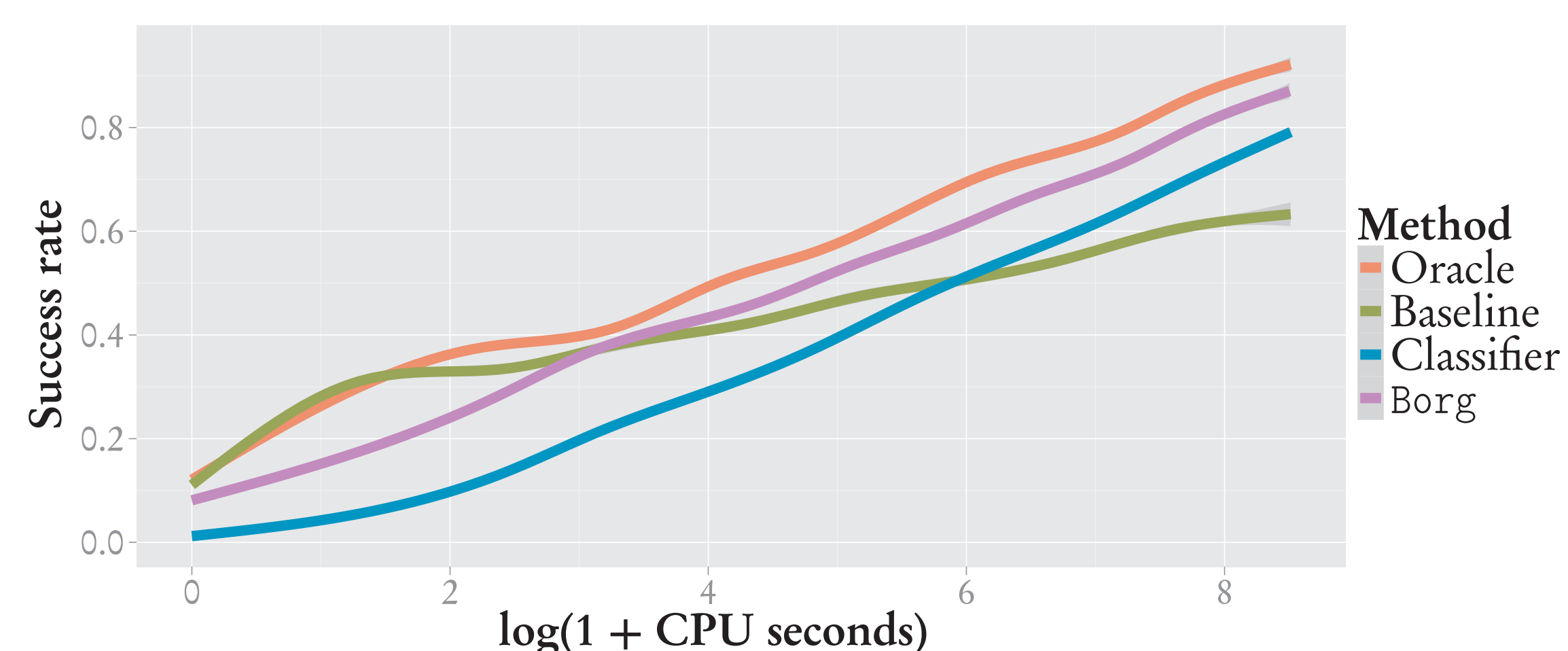
Hybrid generative/discriminative latent class model

Building on earlier modeling work [1], borg predictions are conditioned both on *static instance features*, like those used by SATzilla, and on the *failure to date* of the solvers already executed. The full model is built from three sets of components:

- ▶ “run classes”, which independently model solvers’ typical runtime distributions,
- ▶ “task classes”, which associate the run classes of solvers on typical instances, and
- ▶ regression models of static features of the instances belonging to every task class.

Static instance features narrow down the likely class of an instance, and solver failures provide additional information about both its task class and its run classes.

Experimental results: SAT 2009 instances (random)



Over repeated train / test splits of the SAT 2009 competition “random” category, borg quickly overcomes the overhead of feature computation to provide substantially better performance than both the best single solver and a purely classifier-based algorithm selection portfolio. Its performance comes surprisingly close that of an optimal portfolio with oracle knowledge.

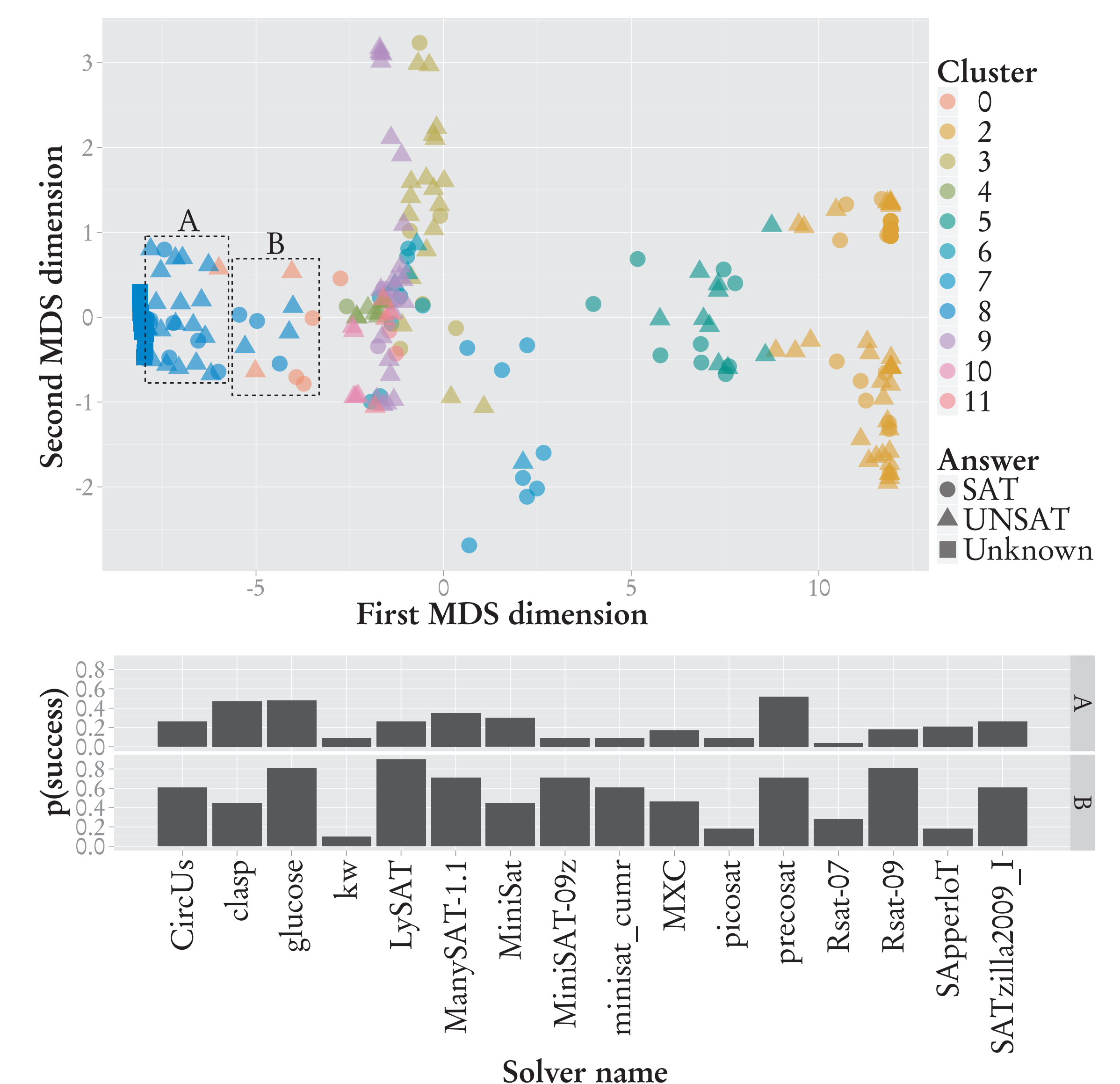
Interactive visualization of solver performance data

Solver competitions have become an indispensable tool for surveying the state of SAT and other fields, but discovering patterns in the data generated by these competitions can be a challenge. The probabilistic modeling work undertaken for borg provides a useful framework for interpreting solver performance data. An experimental web-based visualization tool, linked from

<http://nn.cs.utexas.edu/pages/research/borg/>

has been developed around this framework. It projects the set of competition instances into a two-dimensional space, and provides interactive functionality for plotting the differences in solver performance across regions of that space.

Projection example: SAT 2009 instances (application)



Using the KL divergence between model-predicted runtime distributions as a dissimilarity measure, and multidimensional scaling (MDS) as a projection method, structure is visually apparent in the set of “application” instances at the SAT 2009 competition. Solver performance data are displayed in two regions of interest. The web tool, linked above, similarly allows arbitrary regions of the projected instance space to be interactively selected and plotted.

References

- [1] Bryan Silverthorn and Risto Miikkulainen. Latent class models for algorithm portfolio methods. In *AAAI*, 2010.
- [2] Bryan Silverthorn and Risto Miikkulainen. Predicting polarity from structure in SAT. In *SAT*, 2011.
- [3] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *JAIR*, 2008.