

Latent Class Models for Algorithm Portfolio Methods

Bryan Silverthorn and Risto Miikkulainen

Department of Computer Science
The University of Texas at Austin

Abstract

Different solvers for computationally difficult problems perform best on different problem instances. *Algorithm portfolios* predict solvers' performance on an instance, then shift computational resources to the solvers that appear best suited. Generative models of solver behavior are an appealing approach to performance prediction. Two models are proposed, both following from an assumption that problem instances cluster into latent classes: a mixture of multinomial distributions, and a mixture of Dirichlet compound multinomial distributions. The latter model captures *burstiness*, the tendency of solver outcomes to recur. When used to run standard solvers on competition benchmarks, portfolios based on these mixture models are found competitive despite using minimal domain knowledge. Their success suggests a path to more powerful and more general algorithm portfolio methods.

What is an algorithm portfolio?

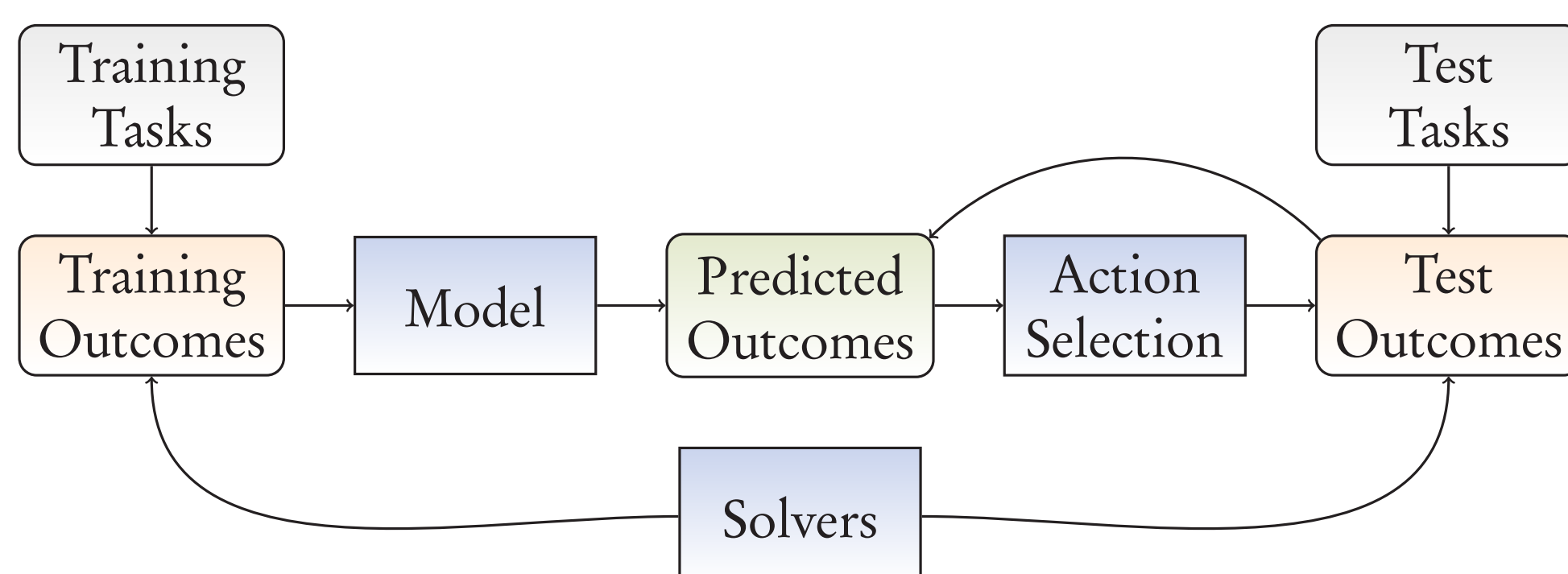
Many **computational problems**, such as satisfiability (SAT), are **intractable** in the worst case. (SAT asks: "does a truth assignment exist that makes a given Boolean expression true?") Heuristic solvers frequently succeed on large instances [2]—but many solvers exist, and **choosing a good solver** for a particular instance is crucial.

An **algorithm portfolio** [1] is the combination of

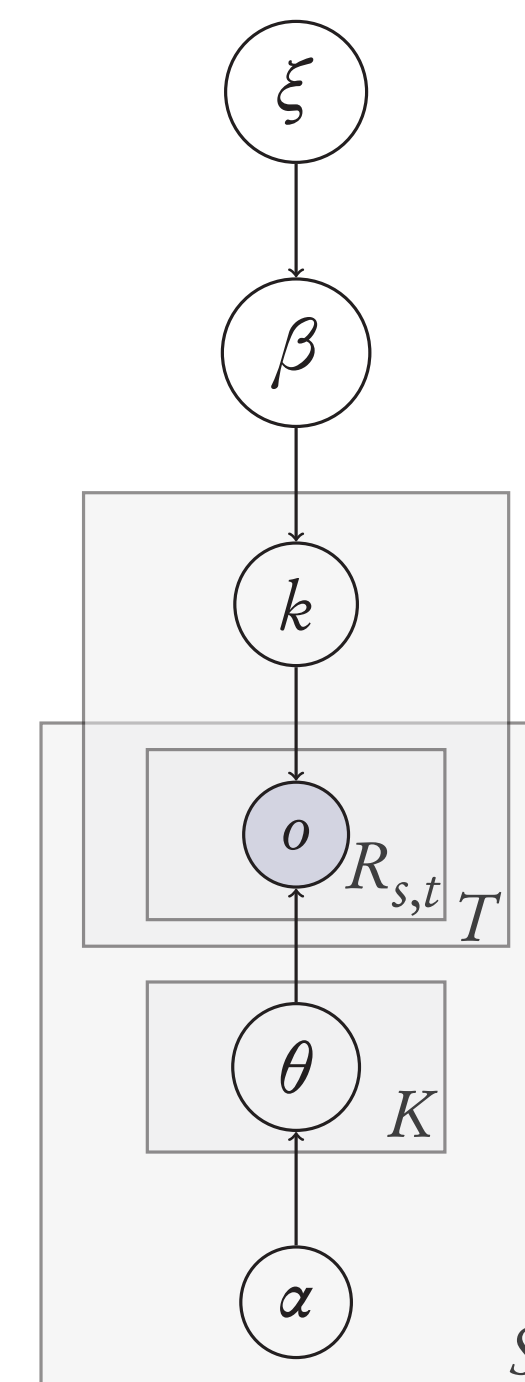
- ▶ a **pool** of algorithms ("solvers"), and
- ▶ a **policy** for scheduling their execution.

The core of an algorithm portfolio is some method for **predicting solver performance** on each task. Classifiers are often used [4], but generative models are an appealing alternative. This work builds **mixture models** of discrete solver outcomes ("proves satisfiability" could be one such outcome) and **evaluates** their effectiveness.

A model-centric portfolio architecture



The multinomial model of solver behavior



Process

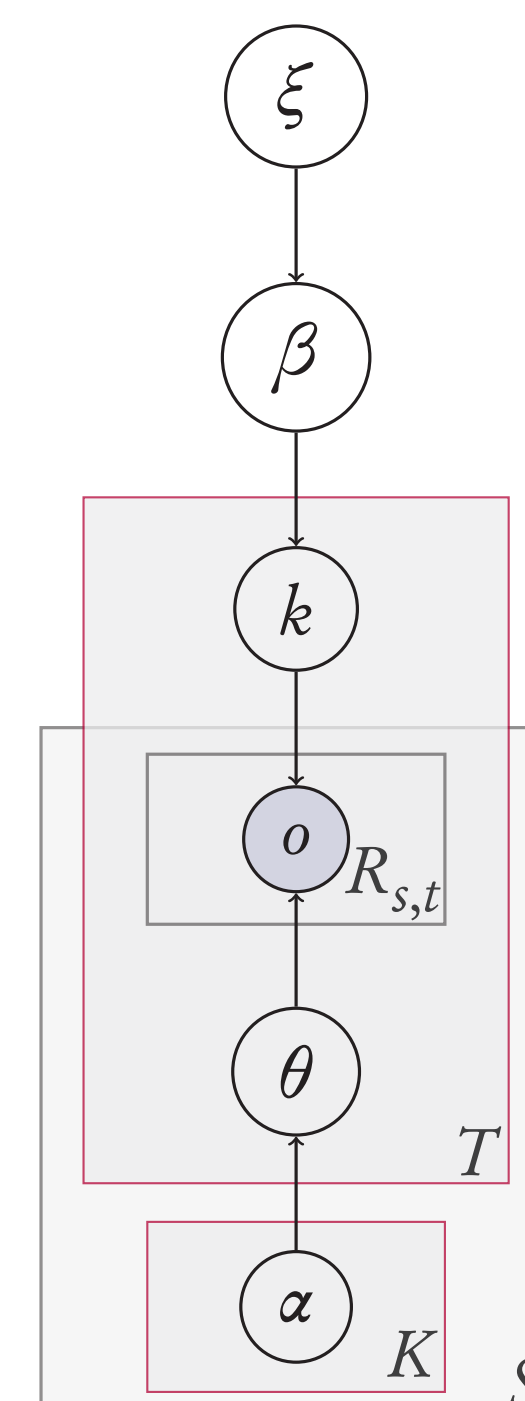
$$\begin{aligned} \beta &\sim \text{Dir}(\xi) \\ k_t &\sim \text{Mult}(\beta) & t \in 1 \dots T \\ \theta_{s,k} &\sim \text{Dir}(\alpha_s) & s \in 1 \dots S \\ o_{t,s,r} &\sim \text{Mult}(\theta_{s,k_t}) & t \in 1 \dots T \\ & & s \in 1 \dots S \\ & & r \in 1 \dots R_{s,t} \end{aligned}$$

Key

ξ class prior α outcome prior
 β class distr. θ outcome distr.
 k class o outcome
 T tasks S actions
 K classes R runs

Solvers and run durations are paired to form **independent actions**. Problem instances—"tasks"—are grouped into latent classes. This clustering **captures similarities** amongst actions and amongst tasks.

The bursty (DCM) model of solver behavior



Process

$$\begin{aligned} \beta &\sim \text{Dir}(\xi) \\ k_t &\sim \text{Mult}(\beta) & t \in 1 \dots T \\ \theta_{t,s} &\sim \text{Dir}(\alpha_{s,k_t}) & s \in 1 \dots S \\ o_{t,s,r} &\sim \text{Mult}(\theta_{t,s}) & t \in 1 \dots T \\ & & s \in 1 \dots S \\ & & r \in 1 \dots R_{s,t} \end{aligned}$$

Key

ξ class prior α outcome root
 β class distr. θ outcome distr.
 k class o outcome
 T tasks S actions
 K classes R runs

The DCM model includes a per-action outcome distribution for **each** task, allowing it to model **burstiness** in outcomes—and capturing, for example, the deterministic behavior of certain solvers.

Action selection with a latent class model

Outcomes of **past actions** narrow the likely class of a task. Under the DCM model, they also provide evidence of that task's outcome distribution. How should the **next action** be chosen? One efficient approach is to choose greedily, maximizing (or randomly selecting, proportional to) the **expected utility** of only the next action. To penalize costly actions, the expected utility of a solver run of c seconds is discounted by γ^c given a discount factor γ .

Experimental results

- The 2009 SAT competition provides a testbed. In cross-validation, we
- ▶ **run** each solver multiple times on each training task;
 - ▶ **fit** model parameters to the outcomes observed; and,
 - ▶ for each test task, until the task is solved or its time limit exceeded, repeatedly **select** an action using a greedy policy and the fitted model.

Method	SAT Instances Solved (σ)		
	random	crafted	indust.
Best Single	320.8 (3.7)	122.6 (3.8)	153.5 (3.2)
Random	261.6 (4.8)	89.3 (3.5)	54.5 (3.4)
SATzilla [4]	407.5 (3.1)	125.6 (3.3)	137.6 (3.3)
Soft + Mult.	319.3 (8.2)	116.9 (4.9)	136.2 (4.3)
Soft + DCM	342.2 (8.2)	118.2 (4.3)	138.1 (5.0)
Hard + Mult.	340.3 (45.7)	104.8 (9.7)	129.4 (8.0)
Hard + DCM	424.2 (12.9)	129.4 (6.5)	146.0 (5.6)

Latent class models appear to model the domain sufficiently well to drive an effective algorithm portfolio [3]. They **offer a foundation** for more comprehensive models and for improved portfolio methods.

Acknowledgments

This research was supported in part by NSF grants EIA-0303609, IIS-0757479, and IIS-0915038, and by THECB grant 003658-0036-2007.

References

- [1] Bernardo Huberman, Rajan Lukose, and Tad Hogg. Economics Approach to Hard Computational Problems. *Science*, January 1997.
- [2] Henry Kautz and Bart Selman. The state of SAT. *DAM*, June 2007.
- [3] Bryan Silverthorn and Risto Miikkulainen. Latent Class Models for Algorithm Portfolio Methods. In *AAAI*, 2010.
- [4] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based Algorithm Selection for SAT. *JAIR*, 2008.