IBM Watson Research Center

Snappy

A Simple Neighborhood-based Algorithm Portfolio in PYthon

Horst Samulowitz, Chandra Reddy, Ashish Sabharwal, Meinolf Sellman
IBM TJ Watson Research Center, New York

Need for an Easy-to-Use Portfolio "Tool"!

- Despite proven success, users of most portfolios are:
  - their own creators, or
  - creators of other portfolios: need comparison to publish!
- Not yet adopted by SAT/CSP/MIP communities at large
  - "portfolio builders" for some are not available; others require significant familiarity with packages such as MATLAB
  - could allow showcasing the benefit of heuristics that are great on some instances but not necessarily in overall average performance!
- **Goal**: Create a Portfolio Solver that is:
  - Easy to Use by non-portfolio-creators
    - Must run "out-of-the box" without the need for an offline "training phase"
    - Can possibly improve its performance through Online Learning
  - Easy to Understand, Parse, and Extend
  - Has Competitive Performance across a Variety of Domains

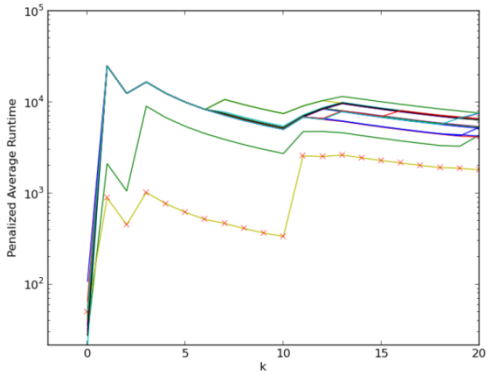
Snappy: A Simple Portfolio in Python

[SAT-2013]

- **Available: http://researcher.watson.ibm.com/researcher/files/us-samulowitz/snappy.zip**
- Readable: one single Python script, using scipy and other standard libraries
- Based on k-Nearest Neighbor performance information
  - Various metrics: Euclidean, Canberra, Mahalanobis, Minkowski, …
- Needs Prior Performance data, as usual
- BUT No "Training" Phase: solver selection based on an "easy" function of the performance of all algorithm on k neighbors of test instance:
  - E.g., "min min" aggregation function:

$$\arg \min_{A \in \text{Algorithms}} \min_{k \in [k_{min}, k_{max}]} \text{PAR}(A, k \text{ nearest neighbors of the test instance})$$

- "execution" mode: run on a new test instance
- "analysis" mode: run on several train-test splits, produce insights through summaries, charts, etc.
  - Can be tuned and improved, if desired



Snappy: Out-of-the-Box Performance

- Competitive simultaneously on several benchmarks with *a single default setting!* (3S and SATzilla trained for each benchmark row)

Benchmark				3S		snappy	
Name	#Alg	#Feat.	Timeout	%	PAR-1	%	PAR-1
SAT-2011-splits	37	48	5000	91.23	772.8	94.52	512.5
SAT-2012-10fold-f1	72	48	2000	96.59	174.3	96.48	161.8
SAT-2012-comp-f1	72	48	2000	83.05	556.4	83.77	560.5
SAT-2012-10fold-f2	72	32	2000	97.23	146.1	96.17	167.5
SAT-2012-comp-f2	72	32	2000	85.42	499.1	85.42	526.3

Benchmark				SATzilla		snappy	
Name	#Alg	#Feat.	Timeout	%	PAR-1	%	PAR-1
Industrial	18	125	5000	75.3	1685	72.6	1789
Crafted	15	125	5000	66.0	2096	63.3	2198
Random	9	125	5000	80.8	1172	80.3	1221

Based on results reported by Xu et al [SAT-2012]

Last years tool

- **SatX10:** Plug&Play Parallel SAT Solver
  - Supports information sharing, configurable communication patterns
  - Framework available since last year
  - With Competition: Version available that has several solvers integrated already (e.g., Glucose, CircMinisat, MiniSat, etc.)
  - Should run on 1 to 10,000 cores