



# AUTOMATIC ALGORITHM CONFIGURATION – APPLICATIONS

---

Marcus Ritt

May 2017

# Introduction and Overview

---

- Present some successful applications of AAC.
- Discussion and critique.

---

Problem	What	How	Ref.
TSP/QAP	GA (Sel,Mut,Xover)	GP	Oltean (2005)
Timetabling	Heuristic alg.	Race	Chiarandini et al. (2006)
SAT	Var. sel. rule (LS)	GE	Fukunaga (2008)
1D bin packing	Perturb. (IGA)	GE	Burke et al. (2012)
VRP	Perturb. (VNS)	GE	Drake et al. (2013)
Flow shop	Perturb. (IGA)	Race	Mascia et al. (2013)
Job shop	Disp. rules (Greedy)	GE	Nguyen et al. (2015)
SAT	SLS (Var. sel.)	P-ILS	KhudaBukhsh et al. (2016)

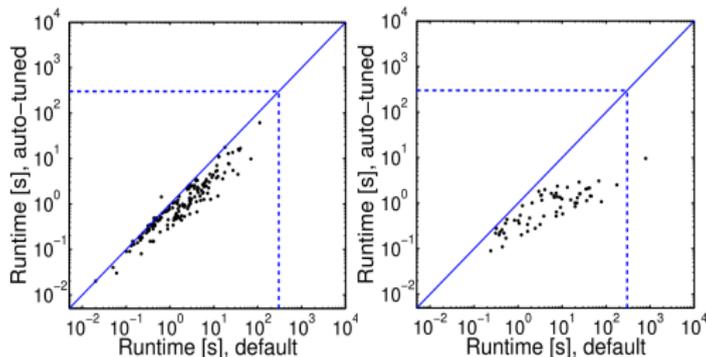
---

**Example: parameter tuning**

---

## Tuning CPLEX 10.1.1 (Hutter 2009)

- **Parameters:** 50 categorial, 8 integer, and 5 real.
- **Test problems:** about 5K, including machine-job assignment (MJA), mixed integer knapsack (MIK), QP.
- 48 h configuration, 300 s cutoff time per run.
- Min. **average runtime**, cutoff penalty 10, with FocusedILS.
- Speedups: 2 (MJA) to 23 (MIK).



CPLEX-MJA. 5.37s vs 2.395s

CPLEX-MIK. 28s vs 1.2s

Source: Hutter (2009), cf. <https://www.youtube.com/watch?v=bA-XwDStjng>

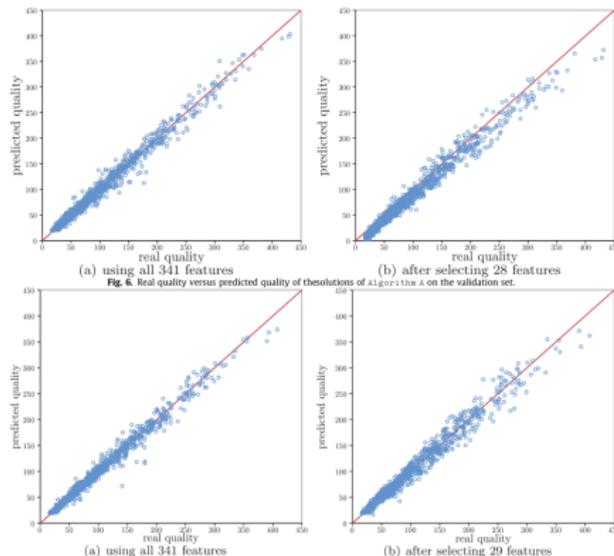
**Example: algorithm selection  
(portfolios)**

---

- Precedence ordered set of **tasks**  $J$ , with processing time  $p_j$ .
- $K$  **resources** with limits  $R_k$ , some renewable other non-renewable.
- For each task  $M_j$  modes, and **mode-dependent resource consumption**  $r_{jkm}$ .
- Goal: find a schedule of **minimal makespan**.

- 686 instance features related to
  - **Size**: e.g. number of activities, number of resources.
  - **Resources**: summary statistics over activities.
  - **Activities**: summary statistics over resources.
  - **Precedences**: e.g. order strength, degrees.
  - **Processing times**: summary statistics.
- **Elimination** of 103 constant and 242 correlated features on 3140 instances.
- Two **algorithms** A (Tabu search) and B (hybrid genetic algorithm).
- **Performance**: average value of 5 replications after fixed no. of schedule evaluations (5K, 25K).

- Bi-directional correlation based *feature selection*:  $\approx 20\text{--}30$  features.
- Among ML methods *M5P-tree* (decision tree with linear model leafs) and *M5-rules* (decision rules for linear models) best.



- Two approaches:
  - AS1: choose algorithm with *best performance* prediction.
  - AS2: learn *best choice* directly, best ML method: random forests.

Schedules		A	B	AS1	AS2	AS
5K	Corr. classified (%)	58.1	49.9	69.0	83.2	100
	ARD (%)	2.30	2.85	1.32	0.51	0.00
	Avg. makespan	100.68	99.40	98.51	97.74	97.34
25K	Corr. classified (%)	89.6	18.0	80.1	93.3	100
	ARD (%)	0.36	8.01	1.04	0.18	0.00
	Avg. makespan	92.25	99.98	92.66	92.02	97.34

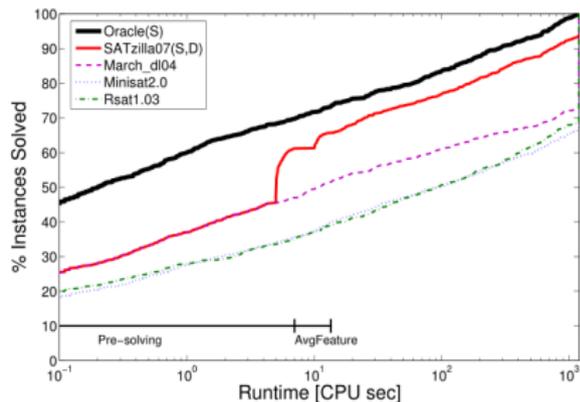
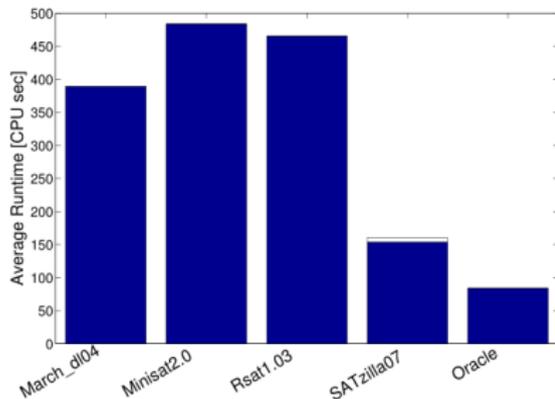
- ***Pre-processing***
  - Identify features, construct empirical performance models.
  - Select 2 pre-solvers, a set of main solvers, and a backup solver.
- ***Solving***
  - Pre-solvers run with very limited budget to filter easy instances.
  - Then select main solver: best algorithm based on instance features.
  - If selection fails: use backup solver.
  - If main solver fails: use next best solver.

## Feature and performance models

- **Features**
  - 84 features from Nudelman et al. (2004).
  - Greedy forward selection of single, then quadratic features.
- **Performance model**
  - Feature matrix  $\Phi \in \mathbb{R}^{n \times d}$ ,  $n$  test instances,  $d$  features, vector of  $y_i = \log r_i$  of log-runtime.
  - Linear model  $\Phi w = y$ ,  $w$  determined by Ridge regression.
  - Performance predictor:  $r = \exp(w^t f)$  for instance with features  $f$ .
  - Special handling of censored data (cutoff runs).
  - Model improved by separation into satisfiable and unsatisfiable instances.

# 3

## Results (instances from ALL categories)



**Example: automatic algorithm  
configuration**

---

- Configurable *stochastic local search* SAT solver.
- *Diversification*: clause selector, variable selector, probability.
- *Intensification*: choice between WalkSAT-like algorithms, dynamic local search algorithms, and G<sup>2</sup>WSAT-like algorithms.
- Rigid, finite search space (compared to e.g. Fukunaga (2008)).
- 7 d configuration, 5 s cutoff time per run.
- Min. *average runtime*, cutoff penalty 10, with FocusedILS.

## Results (example)

Performance of SATenstein-LS solvers, the best challengers with default configurations and the best automatically configured challengers. Every algorithm was run 25 times on each instance with a cutoff of 600 CPU seconds per run. Each table entry  $(i, j)$  indicates the test-set performance of algorithm  $i$  on distribution  $j$  as  $a/b/c$ , where  $a$  (top) is the penalized average runtime;  $b$  (middle) is the median of the median runtimes over all instances;  $c$  (bottom) is the percentage of instances solved (i.e., those with median runtime < cutoff).

Distribution	QCP	SW-GCP	R3SAT	HGEN	FAC	CBMC(SE)
Best Challenger (default)	ANOV	RANOV	PAWS	RSAPS	SAPS	VW
Performance	25.42 0.02 99.6%	0.15 0.12 <b>100%</b>	1.77 <b>0.08</b> <b>100%</b>	33.28 2.33 99.7%	16.41 10.56 <b>100%</b>	385.12 0.23 93.4%
Best Challenger (tuned)	VW[D]	G2[D]	VW[D]	SAPS[D]	SAPS[D]	VW[D]
Performance	0.33 0.02 <b>100%</b>	0.05 0.05 <b>100%</b>	1.26 0.15 <b>100%</b>	31.77 0.75 99.6%	<b>10.68</b> <b>7.00</b> <b>100%</b>	16.45 <b>0.02</b> <b>100%</b>
SATenstein-LS[D]	<b>0.08</b>	<b>0.03</b>	<b>1.11</b>	<b>0.02</b>	10.89	<b>4.75</b>
Performance	<b>0.01</b> <b>100%</b>	<b>0.02</b> <b>100%</b>	0.14 <b>100%</b>	<b>0.01</b> <b>100%</b>	7.90 <b>100%</b>	<b>0.02</b> <b>100%</b>

Source: KhudaBukhsh et al. (2016)

# Perspectives

---

- Although promising, AAC is *not very common*.
- Approaches are *very specific*, with tailored details.
- It seems often *hard to beat the best single algorithm*.
- The results seem *not robust* wrt to budgets (time, evaluations).



Burke, Edmund K., Matthew R. Hyde, and Graham Kendall (2012). “Grammatical evolution of local search heuristics”. In: *IEEE Trans. Evol. Comp.* 16.2, pp. 406–417. DOI: 10.1109/TEVC.2011.2160401.



Chiarandini, M. et al. (2006). “An effective hybrid algorithm for university course timetabling”. In: *J. Sched.* 9.5, pp. 403–432. DOI: 10.1007/s10951-006-8495-8.



Drake, John H., Nikolaos Kililis, and Ender Özcan (2013). “Generation of VNS Components with Grammatical Evolution for Vehicle Routing”. In: *16th European Conference on Genetic Programming*. Ed. by Krzysztof Krawiec et al. Vol. 7831. LNCS. Vienna, pp. 25–36. DOI: 10.1007/978-3-642-37207-0\_3.

-  Fukunaga, A. S. (2008). “Automated discovery of local search heuristics for satisfiability testing”. In: *Evo. Comput.* 16.1, pp. 31–61. DOI: 10.1162/evco.2008.16.1.31.
-  Hutter, Frank (2009). “Automated configuration of algorithms for solving hard computational problems”. PhD thesis. University of British Columbia, Department of Computer Science.
-  KhudaBukhsh, A. R. et al. (2016). “SATenstein: Automatically building local search SAT solvers from components”. In: *Artif. Intell.* 232, pp. 20–42. DOI: 10.1016/j.artint.2015.11.002.



Mascia, Franco et al. (2013). “From Grammars to Parameters: Automatic Iterated Greedy Design for the Permutation Flow-Shop Problem with Weighted Tardiness”. In: **7th International Conference on Learning and Intelligent Optimization**. Vol. 7997. LNCS, pp. 321–334. DOI: 10.1007/978-3-642-44973-4\_36.



Messelis, Tommy and Patrick De Causmaecker (2014). “An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem”. In: **Eur. J. Oper. Res.** 233, pp. 511–528. DOI: 10.1016/j.ejor.2013.08.021.

-  Nguyen, S. et al. (2015). “Automatic Programming via Iterated Local Search for Dynamic Job Shop Scheduling”. In: *IEEE Trans. Cyber.* 45.1, pp. 1–14. DOI: [10.1109/TCYB.2014.2317488](https://doi.org/10.1109/TCYB.2014.2317488).
-  Nudelman, E. et al. (2004). “Understanding random SAT: Beyond the clause-to-variables ratio”. In: *Tenth International Conference on Principles and Practice of Constraint Programming*, pp. 438–452.
-  Oltean, M. (2005). “Evolving evolutionary algorithms using linear genetic programming”. In: *Evo. Comput.* 13.3, pp. 387–410.
-  Xu, L. et al. (2008). “SATzilla: portfolio-based algorithm selection for SAT”. In: *J. Artif. Intell. Res.* 32, pp. 565–606.