How to Win Gold at a SAT Competition Without Writing a SAT Solver

Serdar Kadıoğlu, Yuri Malitsky, and Meinolf Sellmann

Our paper on "Algorithm Selection and Scheduling" [10] was published at CP 2011, but the origins of this work go back to 2008 when Meinolf and Yuri began working on instance-specific algorithm configuration [5]. Our research showed quickly that employing solver parameterizations at runtime which had never been tested during training was a brittle approach that exhibited too much variance to be applicable in practice. We therefore resorted to an approach that generated a small set of parameterizations during training and then selected one of them at runtime. The ultimate result of the work leading to the domination of the SAT competitions for three years using only outdated solvers which by themselves had never even placed. What the research demonstrated, is the folly of creating jack-of-all trades methodologies. Instead, one needs to create parameterized methodologies and let algorithms tune them and decide when best to use them.

In [5], the research began by clustering training instances, and generating one parameterization for each cluster that worked well for the respective instances. At runtime, we then computed the distances to the cluster centers to determine the nearest cluster - and which corresponding parameterization to run. In other words, we built a small portfolio of algorithms during training, and used a simple Voronoi diagram to map instance features to algorithms to select the parameterization at runtime.

While this approach worked well for us, we were not using a state-of-the-art algorithm selection algorithm. These were based on empirical hardness models [16] at the time. That is, to select the right solver for an instance, the best approach was to forecast each solver's runtime based on instance features, and then select the solver with the lowest expected runtime. The performance forecast was typically based on a parametric model.

We wanted to improve our instance-specific algorithm configurator by augmenting it with a performance-model based algorithm selector. To our surprise, this worsened performance. We needed to check if this was due to the fact that we had specifically generated parameterizations for the clusters, or whether clustering-based algorithm selection was also a valid approach for building algorithm portfolios in general (i.e. for portfolios of algorithms we are given, rather than parameterizations specifically generated).

Running experiments on SAT, where the approach from [16] had excelled before, showed clearly that cluster-based algorithm selection worked better than empirical hardness models! We therefore widened our target: We no longer just wanted a better instance-specific algorithm configurator, we now also wanted a better algorithm portfolio generator.

The first modification we made to the clustering approach for algorithm selection was to center the clusters around test instances. That is, instead of creating fixed clusters upfront, we would create a cluster at runtime, when we knew the features of the test instance. That approach is of course better known as k-nearest neighbors. However, we modified the standard k-NN approach a bit: We would not pick the solver that was the best solver on most instances in the neighborhood. Instead, we chose the solver that would need the shortest amount of time to solve all instances in the cluster. To our knowledge, this created the first cost-based classification algorithm for algorithm selection.

The next step towards a superior portfolio builder was to not just pick one solver, but to generate a schedule of solvers. An offline schedule had previously been proposed in [13]. Our idea was to generate a schedule of solvers at runtime, one that would work best for the k nearest instances in the training set. Since the time needed to build this schedule was directly reducing the time to solve the actual problem instance, we needed a very fast scheduler. Rather than solving the resulting NP-hard set-covering problem with an extremely large number of sets perfectly, we resorted to a column-generation approach at the root to greatly reduce the number of sets. This method worked in seconds and returned efficient solver schedules.

The fast scheduling approach allowed us to generate schedules at runtime, specifically geared towards the instance at hand. However, in preparation for the SAT Competition 2011 [2], we found that fully dynamic schedules tended to over-optimize the schedules towards the nearest neighbor clusters. Therefore, in the entry that won two gold, two silver, and three bronze medals at the competition, we used a static schedule for the first 10% of the available time to catch instances that were easy for at least one high-performing solver. After the schedule was run, we would then select one long-running solver for the remaining 90% of the available time.

Due to the competition rules, the SAT solvers we used in the 2011 competition were all from 2009 or earlier. Note what this means: Just by carefully selecting the best solver for a given instance, while hedging the odds a bit by employing a quick schedule upfront, we could make up for *two years of development time* in what was probably the fastest developing community on combinatorial search algorithms at the time. This is a remarkable testament to the power of algorithm portfolios.

Our paper also had a profound impact on Serdar's professional career and has shaped the research directions he pushed at Oracle. As a member of the Advanced Constraint Technology team, Serdar helped other Oracle products to solve their large-scale combinatorial optimization problems. When faced with complex business requirements under severe runtime constraints, Hybrid Optimization and Decomposition methods such as column generation and logic-based benders' decomposition [3] allow us to combine the complementary strengths of different paradigms. Thanks to the work on this paper, Serdar was able to lead this line of research at Oracle with confidence. In particular, he addressed several business-critical problems with immediate practical impact in Supply-Chain Management [8], Planning & Scheduling [14, 9], Product Configuration [15], and Resource Allocation [7, 6]. These contributions were highlighted by Business Insider [4].

Later, we extended our method for parallel algorithm portfolios which included parallel SAT solvers and won two gold medals at the 2013 SAT Competition as well [11]. At this point, the SAT Competition organizers became hostile enough towards portfolios that the technology was banned altogether. The official reason: To ensure that only a "pure" SAT solvers should be able to claim being best performing SAT Solver (even when this was factually not so). Using instance-specific algorithm tuning, and a new portfolio builder called cost-sensitive hierarchical clustering [12], we nevertheless continued to win seventeen first places at the MaxSAT Evaluations between 2013 and 2016.

Finally, in 2018 we applied automatic algorithm configuration to portfolio building itself [1]. This showed that different sets of solvers and instances favored different portfolio strategies. For some, it was better to have long static schedules, for others it was better to choose only one long running solver, and again for others a dynamic schedule at runtime was best. Thankfully, users do not need to figure this out for themselves, they can simply let the automatic portfolio configurator do its work and receive a portfolio of solvers that is likely years ahead of the performance that any "pure" solver will achieve.

References

[1] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. Self-configuring cost-sensitive hierarchical clustering with recourse. In *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, pages 524–534, 2018.

[2] SAT Competition, 2011. http://www.cril.univ-artois.fr/SAT11/phase2.pdf.

[3] J.N. Hooker. Logic-based benders decomposition. *MATHEMATICAL PROGRAMMING*, 96:2003, 1995.

[4] Business Insider. Oracle rock-star engineers changing the company. 2016. Available online at "`https://www.businessinsider.com/oracle-rock-star-engineers-2016-3#serdar-kadioglu-turning-theory-into-products-21`", last accessed on Sept. 27, 2019.

[5] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC – Instance-Specific Algorithm Configuration. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10)*, volume 215 of *Frontiers in Intelligence and Applications*, pages 751–756, 2010.

[6] Serdar Kadioglu. Column generation for interaction coverage in combinatorial software testing. *CoRR*, abs/1712.07081, 2017.

[7] Serdar Kadioglu. Core group placement: allocation and provisioning of heterogeneous resources. *EURO J. Computational Optimization*, 7(3):243–264, 2019.

[8] Serdar Kadioglu, Mike Colena, Steven Huberman, and Claire Bagley. Optimizing the cloud service experience using constraint programming. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, pages 627–637, 2015.

[9] Serdar Kadioglu, Mike Colena, and Samir Sebbah. Heterogeneous resource allocation in cloud management. In *15th IEEE International Symposium on Network Computing and Applications, NCA 2016, Cambridge, Boston, MA, USA, October 31 - November 2, 2016*, pages 35–38, 2016.

[10] Serdar Kadioglu, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm selection and scheduling. In *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, pages 454–469, 2011.

[11] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Parallel SAT solver selection and scheduling. In *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 512–526, 2012.

[12] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 608–614, 2013.

[13] Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan. Using case-based reasoning in an algorithm portfolio for constraint solving? *Irish Conference on Artificial Intelligence and Cognitive Science*, 03 2013.

[14] Samir Sebbah, Claire Bagley, Mike Colena, and Serdar Kadioglu. Availability optimization in cloud-based in-memory data grids. In *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, pages 666–679, 2016.

[15] Siebel. Crm innovation pack: Product configurator and oracle advanced constraint technology. *Advanced Constraint Technology (ACT)*, 2016. Available online at "`https://support.oracle.com/knowledge/Siebel/2112562_1.html`", last accessed on Sept. 27, 2019.

[16] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res.*, 32:565–606, 2008.