Automatic Tuning of Solvers

Carlos Ansotegui, Meinolf Sellmann, and Kevin Tierney

This paper is a good example for how long some research ideas can simmer before they are realized. The first time the idea to build a tool for tuning constraint solvers automatically was discussed was between Warwick Harvey and Meinolf during CP 2003 in Kinsale (Ireland). Symmetry breaking was a hot topic at the conference, but it, like many other techniques in CP, seemed to require careful solver calibration to work well. The general question arose to what extent experimental results actually allowed us to compare different algorithmic approaches when performance depends so heavily on parameter settings. A tool was needed to tune solvers automatically to level the playing field for all competing approaches.

It took until 2007 until we revisited the idea. Meinolf and his graduate student Yuri Malitsky followed up on an intriguing observation made by Egon Balas [6], who found that randomly mixing heuristics in a greedy algorithm for set covering frequently led to better results than sticking to the best pure heuristic. This led to the following question:

> Could one have the machine learn automatically which instance-specific probabilities to use for a set of branching heuristics in a randomized branch-and-bound tree search?

Extracting instance features, building a supervised training set, and then inferring probabilities for each heuristic by multinomial logistic regression worked better than using uniform randomization, but left lots of room for improvement. The main problem with this straight forward ML approach consists in the inherent chicken-and-egg problem when branching: On the one hand, how we branch determines which sub-problems we will need to branch on going forward. On the other hand, the distribution of sub-problems is needed to build a representative training set for learning how to branch. Curiously, the effect of branching on the subproblem distribution can lead to a virtual improvement of search heuristics that learn absolutely nothing [8]. This obviously makes it impossible to create a traditional ML training set effectively.

To deal with the chicken-and-egg problem, a holistic tuning approach is needed that considers the entire search method. The alternative approach that Yuri and Meinolf therefore invented was to run a continuous local search over the continuous parameters of the search for clusters of instances, which worked surprisingly well [9].

In the summer of 2008, Carlos visited Meinolf at Brown during his sabbatical. They took a trip to Point Judith Beach in Rhode Island, together with Yuri

and Meinolf's other grad student Serdar Kadioglu. (On the way to the beach, they were passed by the Brown CS Department Chair. Much to Meinolf's relief, the Chair did not recognize them as he zipped by on the highway.) At the beach, we discussed what general search method might be best suited for parameter tuning. A robust method was needed that could handle categorical, ordinal, and continuous variables, an approach that works well without gradients and for rugged objective landscapes, and that offered inherent capabilities for parallelization. We wondered if it was possible to use genetic programming for building a SAT solver by evolving the code through parameter configuration. And so the idea to use an evolutionary algorithm for parameter tuning was born.

Shortly afterwards, Kevin started as a Masters student at Brown and chose the project of implementing a genetic algorithm for parameter tuning. His experiments quickly showed that the general evolutionary approach failed to work well when few objective evaluations were affordable. This analysis then led to the gender separation of the approach we finally submitted to CP, thus leading to the gender-based genetic algorithm (GGA) [2] that is still one of the most effective approaches for parameter tuning to date, with recent extensions allowing it to be distributed across a high-performance cluster and use a learned model for creating new parameter settings [3].

Ten years after the original inception, one of the most exciting uses of GGA is hyper-reactive search (HRS). In essence, the idea is to use runtime features of a search algorithm (such as the number of steps since we last saw an improving solution) to guide how the search continues. Here, GGA "learns" how the runtime features should translate into stochastic search decisions to get the best results. The approach was applied to Tabu Search [4] and Dialectic Search [7, 1] for MaxSAT and won four categories at the 2016 MaxSAT Evaluation [5]. HRS has revived the vision of a true meta-heuristic search approach that can self-adapt and works without heavy manual tuning: The winning MaxSAT solver took a mere two days to implement and literally knows nothing more about MaxSAT than how to evaluate the objective. The strength of the approach results purely from allowing the machine to learn from experience how to conduct a search more effectively.

References

[1] C. Ansótegui, J. Pon, M. Sellmann, and K. Tierney. Reactive dialectic search portfolios for maxsat. In *AAAI*, pages 765–772, 2017.

[2] C. Ansotegui, M. Sellmann, and K. Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In I.P. Gent, editor, *Principles and Practice of Constraint Programming (CP-09)*, volume 5732 of *LNCS*, pages 142–157. Springer, 2009.

[3] Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney. Model-based genetic algorithms for algorithm configuration.

In *24th International Joint Conference on Artificial Intelligence*, pages 733–739, 2015.

[4] C. Ansótegui, B. Heymann, J. Pon, M. Sellmann, and K. Tierney. Hyper-reactive tabu search for maxsat. In *International Conference on Learning and Intelligent Optimization*. Springer, 2018.

[5] J. Argelich, C.M. Li, F. Manyà, and J. Planes. MaxSAT Evaluation, 2016. www.maxsat.udl.cat.

[6] E. Balas and M. C. Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890, 1996.

[7] Serdar Kadioglu and Meinolf Sellmann. Dialectic search. In *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, pages 486–500, 2009.

[8] D. H. Leventhal and M. Sellmann. The accuracy of search heuristics: An empirical study on knapsack problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008, Paris, France, May 20-23, 2008, Proceedings*, pages 142–157, 2008.

[9] Y. Malitsky and M. Sellmann. Stochastic offline programming. In *ICTAI 2009, 21st IEEE International Conference on Tools with Artificial Intelligence, Newark, New Jersey, USA, 2-4 November 2009*, pages 784–791, 2009.