Self-adaptive Systems: Facilitating the Use of Combinatorial Problem Solvers

Broderick Crawford[1,2], Ricardo Soto[3,4], Eric Monfroy[5], and Franklin Johnson[6]

[1] Universidad San Sebastián, Chile
[2] Universidad Finis Terrae, Chile
[3] Pontificia Universidad Católica de Valparaíso, Chile
[4] Universidad Autónoma de Chile, Chile
[5] CNRS, LINA, Université de Nantes, France
[6] Universidad de Playa Ancha, Chile
broderick.crawford.l@gmail.com,
ricardo.soto@ucv.cl,
eric.monfroy@univ-nantes.fr,
franklin.johnson@upla.cl

Abstract. New methods in Combinatorial Problem Solving can solve larger problems in different domains. They also became more complex, which means that they are hard to use and fine-tuning to the peculiarities of a given problem, limiting its use to a small set of experts, and instead black-box solvers with automated search procedure are needed for its broad applicability. Autonomous Search Systems represent a new research field defined to precisely address the above challenge. The main goal of this paper is to review recent works on this kind of Self-adaptive Systems from the standpoint of the actual requirement for solvers.

Keywords: Self-adaptive Systems, Autonomous Search Systems, Combinatorial Problem Solvers.

1 Introduction

An Autonomous Search (AS) system should provide the ability to modify its internal components (heuristics, inference mechanisms, etc.) when exposed to changing external forces and opportunities. As corresponds to an instance of Self-adaptive Systems with the objective of improving its problem solving performance by adapting its search strategy to the problem at hand. Autonomous search is particularly relevant to the Constraint Programming community, where much work has been conducted to improve the efficiency and usability of constraint solvers. AS provides to a system the ability to change its components in order to improve its problem solving performance. AS can be defined as search processes that integrate control in their solving process either by self adaptation or by supervised adaptation [18]. This control allows an AS system to improve its solving performance by modifying and adjusting itself to the problem at hand. In more detail, the notion of control is present when the parameters or heuristics

C. Stephanidis (Ed.): HCII 2014 Posters, Part I, CCIS 434, pp. 503–508, 2014.

are adjusted online, i.e., when the constraint solver is running. Different methods such as control encoding, control variable and value selection, and evolving heuristics have been proposed to provide control during solving [18].

Concerning the control, in self adaptation, techniques are tightly integrated with the search process and usually require some overhead. The algorithm is observing its own behavior in an online fashion, modifying its parameters accordingly. This information can be either directly collected on the problem or indirectly computed through the perceived efficiency of individual components. Because the adaptation is done online, there is an important trade-off between the time spent computing process information and the gains that are to be expected from this information. Therefore we can consider that the most appropriate strategy depends on the set of computed states and changes during solving. Supervised adaptation works at a higher level. It is usually external and its mechanisms are not coupled with the search process. It can be seen as a monitor that observes the search and analyzes it. Then it modifies the components of the solver (or requires the solver to modify its components) in order to adapt it. Supervised adaptation can use more information, e.g., learning-based knowledge.

2 Recent Advances in Constraint Solving

In recent years different efforts of research have been conducted in order to improve the efficiency of solvers. These improvements often rely on new heuristics, adjustement of parameters and heuristics before/during solving and/or hybridization of solving techniques. Diverse domains of research such as parameter setting in evolutionary computing, reactive search, and hyperheuristics have tackled this challenge using different terms and concepts. However, they have common principles and purposes that respond to similar needs.

2.1 Parameter Setting in Evolutionary Computing

The first domain is evolutionary computing, where parameter setting [25] constitutes a major issue and the taxonomy proposed by Eiben et al. [13] may be recalled. Methods are classified depending on whether they attempt to set parameters before the run (tuning) or during the run (control). The goal of parameter tuning is to obtain parameter values that could be useful over a wide range of problems. Such results require a large number of experimental evaluations and are generally based on empirical observations. Parameter control is divided into three branches according to the degree of autonomy of the strategies. Control is deterministic when parameters are changed according to a previously established schedule, adaptive when parameters are modified according to rules that take into account the state of the search, and self-adaptive when parameters are encoded into individuals in order to evolve conjointly with the other variables of the problem.

2.2 Reactive Search

In [2], Reactive Search is characterized by the integration of machine learning techniques into search heuristics. Basically, reactive search allows an internal

flexibility of the solver taking into acount, by learning the past history of the search process. Moreover, an escape mechanism allowing the restart of the system from a new random point is used when the system shows no improvement. In [30] a framework for adaptive enumeration strategies and meta-backtracks for a propagation-based constraint solver has been extended in order to trigger some functions of a solver, or of a hybrid solver to respond to some observations of the solving process. Being able also simply design adaptive hybridisation strategies by just changing some rules of its update component.

2.3 Hyperheuristics

Finding the best configuration of heuristic algorithms is strongly related to the recent notion of Hyperheuristics [5,6,8,10,9]. Hyperheuristics are methods that aim at automating the process of selecting, combining, generating, or adapting several simpler heuristics (or their components) to efficiently solve computational search problems. Hyperheuristics are also defined as *heuristics to choose heuristics* [7] or *heuristics to generate heuristics* [1]. Hyperheuristics that manage a set of given available basic search heuristics by means of search strategies or other parameters have been widely used for solving combinatorial problems.

3 Autonomous Search Mechanisms

A classification of basic search processes has been proposed by Hamadi et al. [18]. It tries to differentiate offline tasks from online processes, tuning (adjustment of parameters and heuristics before solving) from control.

Table 1. Autonomous mechanisms and their strategies

Mechanism	**Strategy**
Tuning before solving [13,25]	Preprocessing [26] Parameter tuning on preliminary experiments [23,36,32] Component setting [16,4,41,21,22]
Control during solving	Control encoding [19] Evolving heuristics [14,17,11] Controlling variable ordering and variale selection in search heuristics [28,3,12,29,20,33] Controlling evaluation function [31,38] Parameter control in metaheuristics algorithms [2,24,34,24,37,39,40,27,15]

As mentioned before, AS has been indeed investigated for many years, across many different areas and under different names. In [18], the way that autonomous mechanisms have been used in the literature are identified. Table 4.1 summarizes the autonomous mechanisms. Autonomous mechanisms are classified in offline tasks (tuning) and online processes (control).

4 Conclusion

This paper has reviewed approaches, methods and challenges in the new field of Autonomous Search systems. The aforementioned approaches are mainly focused on sampling and learning good strategies after solving a problem or a set of problems. We state that an interesting research direction is how we can provide an autonomous solver with an early replacement ("on the fly") of bad-performance strategies without waiting the entire resolution process or an exhaustive analysis of a given class of problems.

References

1. Bader-El-Den, M., Poli, R.: Generating SAT local-search heuristics using a GP hyper-heuristic framework. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 37–49. Springer, Heidelberg (2008)
2. Battiti, R., Brunato, M.: Reactive Search Optimization: Learning while Optimizing. In: Handbook of Metaheuristics, 2nd edn., Springer (2010)
3. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: de Mántaras, R.L., Saitta, L. (eds.) ECAI, pp. 146–150. IOS Press (2004)
4. Boyan, J.A., Moore, A.W.: Learning evaluation functions to improve optimization by local search. Journal of Machine Learning Research 1, 77–112 (2000)
5. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: A survey of hyper-heuristics. Technical Report NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham, Computer Science (2009)
6. Burke, E.K., Kendall, G., Hart, E., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An Emerging Direction inModern Search Technology. In: Handbook of Meta-heuristics, pp. 457–474. Kluwer (2003)
7. Cowling, P., Soubeiga, E.: Neighborhood structures for personnel scheduling: A summit meeting scheduling problem (abstract). In: Burke, E., Erben, W. (eds.) Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling (2000)
8. Cowling, P.I., Kendall, G., Soubeiga, E.: Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoWorkshops 2002. LNCS, vol. 2279, pp. 1–10. Springer, Heidelberg (2002)
9. Crawford, B., Castro, C., Monfroy, E., Soto, R., Palma, W., Paredes, F.: A hyper-heuristic approach for guiding enumeration in constraint solving. In: Schütze, O., Coello Coello, C.A., Tantar, A.-A., Tantar, E., Bouvry, P., Del Moral, P., Legrand, P. (eds.) EVOLVE - A Bridge Between Probability. AISC, vol. 175, pp. 171–188. Springer, Heidelberg (2012)

10. Crawford, B., Soto, R., Monfroy, E., Palma, W., Castro, C., Paredes, F.: Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. Expert Systems with Applications 40(5), 1690–1695 (2013)
11. Dahmani, N., Clautiaux, F., Krichen, S., Talbi, E.-G.: Self-adaptive metaheuristics for solving a multi-objective 2-dimensional vector packing problem. Applied Soft Computing 16, 124–136 (2014)
12. Eén, N., Sörensson, N.: An extensible sat-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
13. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evolutionary Computation 3(2), 124–141 (1999)
14. Epstein, S.L., Freuder, E.C., Wallace, R.J.: Learning to support constraint programmers. Computational Intelligence 21(4), 336–371 (2005)
15. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: Rudolph, et al. (eds.) [15], pp. 175–184
16. Fukunaga, A.S.: Automated discovery of local search heuristics for satisfiability testing. Evolutionary Computation 16(1), 31–61 (2008)
17. Goualard, F., Jermann, C.: A reinforcement learning approach to interval constraint propagation. Constraints 13(1-2), 206–226 (2008)
18. Hamadi, Y., Monfroy, E., Saubion, F.: What is autonomous search? Technical Report MSR-TR-2008-80, Microsoft Research (2008)
19. Hansen, N.: Adaptive encoding: How to render search coordinate system invariant. In: Rudolph, et al. (eds.), pp. 205–214
20. Hu, B., Raidl, G.: Variable neighborhood descent with self-adaptive noeighborhood-ordering. In: Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics (2006)
21. Hutter, F., Hamadi, Y.: Parameter adjustment based on performance prediction: Towards an instance-aware problem solver. Technical Report MSR-TR-2005-125, Microsoft Research, Cambridge, UK (December 2005)
22. Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K.: Performance prediction and automated tuning of randomized and parametric algorithms. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 213–228. Springer, Heidelberg (2006)
23. Hutter, F., Hoos, H.H., Stützle, T.: Automatic algorithm configuration based on local search. In: AAAI, pp. 1152–1157. AAAI Press (2007)
24. Khichane, M., Albert, P., Solnon, C.: An ACO-based reactive framework for ant colony optimization: First experiments on constraint satisfaction problems. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 119–133. Springer, Heidelberg (2009)
25. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. SCI, vol. 54. Springer (2007)
26. Marques-Silva, J., Sakallah, K.A. (eds.): SAT 2007. LNCS, vol. 4501. Springer, Heidelberg (2007)
27. Maturana, J., Saubion, F.: A compass to guide genetic algorithms. In: Rudolph, et al. (eds.), pp. 256–265
28. Mazure, B., Sais, L., Grégoire, É.: Boosting complete techniques thanks to local search methods. Ann. Math. Artif. Intell. 22(3-4), 319–331 (1998)
29. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & OR 24(11), 1097–1100 (1997)
30. Monfroy, E., Castro, C., Crawford, B., Soto, R., Paredes, F., Figueroa, C.: A reactive and hybrid constraint solver. Journal of Experimental and Theoretical Artificial Intelligence 25(1), 1–22 (2013)
31. Morris, P.: The breakout method for escaping from local minima. In: AAAI, pp. 40–45 (1993)

32. Nannen, V., Smit, S.K., E. Eiben, Á.: Costs and benefits of tuning parameters of evolutionary algorithms. In: Rudolph, et al. (eds.), pp. 528–538
33. Puchinger, J., Raidl, G.R.: Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. J. Heuristics 14(5), 457–472 (2008)
34. Randall, M.: Near parameter free ant colony optimisation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 374–381. Springer, Heidelberg (2004)
35. Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.): PPSN 2008. LNCS, vol. 5199. Springer, Heidelberg (2008)
36. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: IEEE Congress on Evolutionary Computation, pp. 399–406 (2009)
37. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Verbeeck, K., Tuyls, K., Nowé, A., Manderick, B., Kuijpers, B. (eds.) BNAIC, pp. 385–386. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten (2005)
38. Thornton, J.: Constraint Weighting Local Search for Constraint Satisfaction. PhD thesis, Griffith University. Australia (2000)
39. Whitacre, J.M., Pham, Q.T., Sarker, R.A.: Credit assignment in adaptive evolutionary algorithms. In: GECCO, pp. 1353–1360. ACM (2006)
40. Wong, Y.-Y., Lee, K.-H., Leung, K.-S., Ho, C.-W.: A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. Soft Comput. 7(8), 506–515 (2003)
41. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for sat. J. Artif. Intell. Res (JAIR) 32, 565–606 (2008)