Adaptive and Multilevel Approach for Constraint Solving

Claudio León de la Barra[1], Broderick Crawford[1,2], Ricardo Soto[1,3], and Eric Monfroy[4]

[1] Pontificia Universidad Católica de Valparaíso, Chile
[2] Universidad Finis Terrae, Chile
[3] Universidad Autónoma de Chile, Chile
[4] CNRS, LINA, Université de Nantes, France
{claudio.leondelabarra,broderick.crawford,ricardo.soto}@ucv.cl,
eric.monfroy@univ-nantes.fr

Abstract. For many real world problems, modeled as Constraint Satisfaction Problems, there are no known efficient algorithms to solve them. The specialized literature offers a variety of solvers, which have shown satisfactory performance. Nevertheless, despite the efforts of the scientific community in developing new strategies, there is no algorithm that is the best for all possible situations. Then, several approaches have emerged to deal with the Algorithm Selection Problem. Here, we sketch the use a Choice Function for guiding a Constraint Programming solver exploiting search process features to dynamically adapt it in order to more efficiently solve Constraint Satisfaction Problems. To determine the best set of parameters of the choice function, an upper-level metaheuristic is used. The main novelty of our approach is that we reconfigure the search based solely on performance data gathered while solving the current problem.

Keywords: Algorithm Selection Problem, Constraint Solving, Constraint Satisfacion Problems, Autonomous Search.

1 Introduction

Optimization problems can be solved by different algorithms, with varied performance for different problem characteristics. Although some algorithms are better than others on average, there is not the best algorithm for all the possible instances of a given problem. To address this concern, recent work has focused on creating algorithm portfolios, which contain a selection of state of the art algorithms. To solve a particular problem with this portfolio, a pre-processing step is run where the suitability of each algorithm for the problem at hand is assessed. It is the same in Constraint Programming (CP), where the selection of an enumeration strategy is crucial for the performance of the resolution process, a correct selection can dramatically reduce the computational cost of finding a solution. However, it is well-known that deciding a priori the correct heuristic

C. Stephanidis (Ed.): Posters, Part I, HCII 2013, CCIS 373, pp. 650–654, 2013.

is quite difficult, as the effects of the strategy can be unpredictable. In recent years, different efforts have been done to determine good strategies based on the information generated through the resolution process. However, deciding what information must be measured and how to redirect the search is an ongoing investigation [1].

Here, we propose a hyperheuristic approach to manage a portfolio of enumeration strategies. A hyperheuristic is a heuristic that operates at a higher level of abstraction than the solver. The hyperheuristic has no problem-specific knowledge, at any given time the hyperheuristic must choose which enumeration strategy to call. To allow the hyperheuristic to operate, we define a choice function which adaptively ranks the enumeration strategies monitoring indicators of the search process. An analogy can be done with the well-known work methodology called Balanced Scorecard (BSC) [12]. Such a business approach aims at helping organizations to translate the strategy in terms of measures. In that way, the behavior and performance of the organization is driven towards the achievement of the strategic objectives. BSC involves concepts such as indicators, measures, strategic objectives and introduces the concept of "means", which refers to actions and initiatives that must be carried out for achieving the organization objectives. Such concepts can be seen as the inspiration of this research, and they are closely related to the actions to follow once we decide to replace the strategy in the CP solver.

2 Constraint Programming

Constraint Programming is a powerful software technology devoted to the efficient resolution of constraint-based problems. It smartly interbreeds ideas from different domains such as Operations Research, Artificial Intelligence, Graph Theory and Programming Languages. Currently, CP is largely used in different application domains. The principle behind CP is simple: the user states the problem and the system solves it.

The solving process demands two main phases: modeling and search. In the modeling phase, the user expresses the problem as a Constraint Satisfaction Problem (CSP), which in general terms corresponds to a sequence of variables lying in a domain and a set of constraints. In the search phase, the CSP is launched in a solving engine, commonly called *solver*, which is a black box composed of a set of powerful search algorithms. Such algorithms are responsible for finding a solution, that is, a variable-value assignment that satisfies the complete set of constraints.

Both phases are essential for an efficient CSP resolution. From a modeling standpoint, a main element is the language. It provides the expressiveness and the semantics for modeling the problems. From a search point of view, the efficiency of search algorithms is the key. The idea is to build a tree data structure holding the potential solutions and to perform a search process by interleaving two phases: constraint propagation and enumeration.

Constraint propagation can be seen as a filtering mechanism. It is able to prune the search tree by deleting those values that do not lead to any solution.

The enumeration, also called labeling, is responsible for creating the branches of the tree by assigning a value to a variable from its domain. The common idea is to generate one branch for each variable-value assignment until a complete solution is reached. However, if a partial solution violates a constraint, the process backtracks, i.e., it returns to the most recently instantiated variable that still has chance to reach a solution. This phase requires selecting the variable to be enumerated and then the value to be assigned; we refer to these steps as the variable and value selection heuristics.

3 The Enumeration Strategy Challenge

Jointly, a variable selection heuristic and a value selection heuristic constitute what is known as the enumeration strategy [1]. Such a pair of decisions is crucial in the performance of the resolution process, where a correct selection can dramatically reduce the computational cost of finding a solution. For instance, consider the simple case of choosing the right value on the first try for each variable: a solution could be found without performing backtracks.

For a simple CSP problem, a good enumeration strategy goes directly to a solution performing a few enumerations without backtracking. However, a bad strategy can perform a lot of backtracks before reaching a solution. Obviously strategies have drastically different efficiencies, often several orders of magnitude, and thus it is crucial to select a good one that unfortunately cannot be predicted in the general case. We are interested in making good choices for enumeration, i.e., selection of a variable and a value.

There exist various studies about enumeration strategies [3–5], some centered in defining general criteria, e.g., the smallest domain for variable selection, and its minimum, maximum, or a random value for value selection. As opposed to this idea, some research works have proposed strategies for a given class of problems, e.g., for job shop scheduling [14, 15], as well as for configuration design [6]. We can also find research focused on determining the best strategy based in some static criterion [2, 3, 16], i.e., the selection heuristic is determined only once before starting the resolution process, remaining unchangeable during the whole process. However, deciding a priori the correct heuristics is quite hard, as the effects of the strategy can be unpredictable.

We proposed techniques allowing the identification and measurement of indicators for the resolution process. The main goal is to make possible the classification of the execution process state, considered as the resolution progress, and in that way be able to determine if the current strategy exhibits a poor performance and whether it is necessary to replace it with a better one. Such an evaluation procedure is not carried out for improving the resolution of a single problem. We address our approach to efficiently find solutions for different problems. This can be done by exploiting search process features to dynamically adapt a CP solver changing the enumeration strategy in use when the other strategy looks more promising in order to solve the CSP at hand. The main novelty of our approach is that we reconfigure the searching based solely on performance data gathered

while solving the current problem performing a general, correct, and opportune indicator-based detection. Becoming our solver in an Autonomous Search (AS) system [11].

4 Adaptive and Multilevel Approach for Constraint Solving

As mentioned above, this work addresses dynamic selection of enumeration strategies for solving constraint satisfaction problems. We focus our research in reacting on the fly, allowing an early replacement of bad performance strategies without waiting the entire solution process or an exhaustive analysis of a given class of problems. We make profit of indicators that gather performance data during the resolution process. Regarding this issue, we use AS mechanisms where a choice function adaptively ranks the enumeration strategies and the problem of determining the best set of parameters of the choice function is tackled using a metaheuristic (Genetic algorithm, Particle swarm ...).

Our solver is able to detect inefficiencies and, as a result, replace the enumeration strategy with a better one. To achieve this goal, we perform an indicator based observation during the solving process. The main purpose of indicators is to proportion the relevant information about the behavior of the resolution process. They must reflect the real state of progress in the problem resolution. In this way, we are able to elaborate a correct judgment about the search performance. To this end, we define simple and quantitative indicators, which can be used different times as well as percentage combinations of them depending on the used techniques and/or the problem to solve.

Our research focuses on developing solvers for CSPs. We are concerned with the design of hybrid resolution approaches including constraint programming and metaheuristics. We have been working on that area during the last years, exploring the different issues –from software engineering and optimization– involved in algorithm design, implementation, tuning and experimental evaluation. The details of our related work are in [7–10, 13].

5 Conclusions

Among the main contributions of our work we can state the design and implementation of a solver that is able to measure the search process (using some basic indicators) in order to perform an on the fly replacement of enumeration strategies (using a portfolio of basic enumeration strategies). The solver is based on enumeration strategies of different natures (based on the size of variable domains, on the number of occurrences of the variables in the constraints) and some indicators on the resolution progress (backtracks, visited nodes, variables fixed, shallow backtracks, deep of the search tree, ...). In our approach the replacement of the enumeration strategies is performed depending on a quality rank (priority), which is computed by means of a choice function fine-tuned by a metaheuristic.

References

1. Apt, K.R.: Principles of Constraint Programming. Cambridge University Press (2003)
2. Beck, J.C., Prosser, P., Wallace, R.J.: Trying again to fail-first. In: Faltings, B.V., Petcu, A., Fages, F., Rossi, F. (eds.) CSCLP 2004. LNCS (LNAI), vol. 3419, pp. 41–55. Springer, Heidelberg (2005)
3. Beck, J.C., Prosser, P., Wallace, R.J.: Variable ordering heuristics show promise. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 711–715. Springer, Heidelberg (2004)
4. Christopher Beck, J., Prosser, P., Wallace, R.J.: Toward understanding variable ordering heuristics for constraint satisfaction problems. In: Fourteenth Irish Artificial Intelligence and Cognitive Science Conference (AICS), pp. 11–16 (2003)
5. Castro, C., Monfroy, E., Figueroa, C., Meneses, R.: An Approach for Dynamic Split Strategies in Constraint Solving. In: Gelbukh, A., de Albornoz, Á., Terashima-Marín, H. (eds.) MICAI 2005. LNCS (LNAI), vol. 3789, pp. 162–174. Springer, Heidelberg (2005)
6. Chenouard, R., Granvilliers, L., Sebastian, P.: Search heuristics for constraint-aided embodiment design. AI EDAM 23(2), 175–195 (2009)
7. Crawford, B., Castro, C., Monfroy, E.: Integration of Constraint Programming and Metaheuristics. In: Miguel, I., Ruml, W. (eds.) SARA 2007. LNCS (LNAI), vol. 4612, pp. 397–398. Springer, Heidelberg (2007)
8. 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, Set Oriented Numerics, and Evolutionary Computation II. AISC, vol. 175, pp. 171–188. Springer, Heidelberg (2012)
9. 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 Syst. Appl. 40(5), 1690–1695 (2013)
10. de la Barra, C.L., Crawford, B.: Fostering Creativity Thinking in Agile Software Development. In: Holzinger, A. (ed.) USAB 2007. LNCS, vol. 4799, pp. 415–426. Springer, Heidelberg (2007)
11. Hamadi, Y., Monfroy, E., Saubion, F.: What is autonomous search? Technical Report MSR-TR-2008-80, Microsoft Research (2008)
12. Kaplan, R.S.: Conceptual Foundations of the Balanced Scorecard. SSRN eLibrary (2010)
13. Monfroy, E., Castro, C., Crawford, B.: Adaptive enumeration strategies and metabacktracks for constraint solving. In: Yakhno, T., Neuhold, E.J. (eds.) ADVIS 2006. LNCS, vol. 4243, pp. 354–363. Springer, Heidelberg (2006)
14. Sadeh, N.M., Fox, M.S.: Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. Artif. Intell. 86(1), 1–41 (1996)
15. Smith, S.F., Cheng, C.: Slack-based heuristics for constraint satisfaction scheduling. In: AAAI, pp. 139–144 (1993)
16. Sturdy, P.: Learning Good Variable Orderings. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, p. 997. Springer, Heidelberg (2003)