Large Neighborhood Search for Robust Solutions for Constraint Satisfaction Problems with Ordered Domains

Jheisson López
University College Cork, School of Computer Science, Ireland
SFI Centre for Research Training in Artificial Intelligence, Ireland

Alejandro Arbelaez
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain

Laura Climent
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain

Abstract

Real-world Constraint Satisfaction Problems (CSPs) are subject to uncertainty/dynamism that cannot be known in advance. Some techniques in the literature offer robust solutions for CSPs, but they have low performance in large-scale CSPs. We propose a Large Neighbourhood Search (LNS) algorithm and a value selection heuristic that searches for robust solutions in large-scale CSPs.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence

Keywords and phrases Constraint Programming, Large Neighbourhood Search, Robust Solutions

Digital Object Identifier 10.4230/LIPIcs.DPCP.2022.9

Funding Science Foundation Ireland. Grant number 18/CRT/6223.

1 Introduction and Background

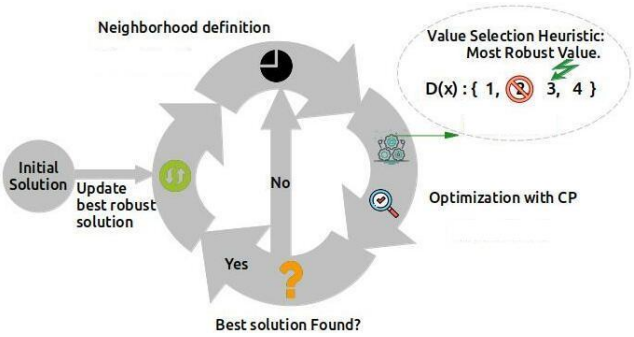
Some real CSPs are dynamic and, even with a high uncertainty about their dynamism, they require solutions that resist changes. The uncertainties may be due to environmental factors, implementation errors or operational failures [1]. In [3], an approach that does not need prior detailed information about the uncertainty of the problem is proposed. The authors assume that future events can restrict the solution space in CSPs with ordered domains and argue that without detailed information about the uncertainty, the most robust solution of a CSP is the one with the maximal distance from all the bounds of the solution space. As in Constraint Programming (CP) the solution spaces can be non-convex, therefore *"we can only ensure that a solution s is located at least at a distance d from a bound in a certain direction of the n-dimensional space if all the possible solutions at distances lower or equal to d from s in this direction are feasible"* [3].

To formalize the previous statements, consider a CSP as a triple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{X} = \{x_1, ..., x_n\}$ is a finite set of variables, $\mathcal{D} = \{\mathcal{D}(x_1), ..., \mathcal{D}(x_n)\}$ is the set of domains of the variables in $\mathcal{X}$ and $\mathcal{C} = \{C_1, C_2, ...C_n\}$ is the set of constraints that must be satisfied. Consider $\mathcal{D}_s(x)$ as the values in $\mathcal{D}(x)$ which are feasible with respect to the possibly partial solution s. If k is an integer parameter which indicate the amount of robustness desired in every variable, $\oplus$ is the set of pair operators $\{\{>, +\}, \{<, -\}\}$ which indicates the feasibility checking directions (e.g. $\oplus_1$ refers to $\{>, +\}$ and $\oplus_{1,1}$ refers to $+$). The set of neighbor values of the v value assigned to x is:

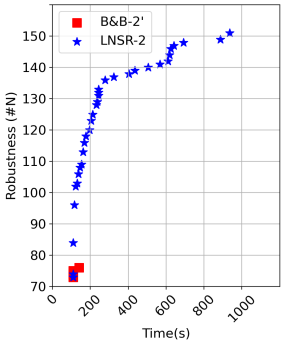
$$\begin{aligned} \mathcal{N}_k(x, v, s, \oplus) \quad &= \{ \quad w \in \mathcal{D}_s(x) : \exists \oplus_i, w \oplus_{i,1} v \wedge |v - w| \leq k \\ \wedge \forall \oplus_z \forall j \in \quad &[1... \quad (|v - w| - 1)], (v \oplus_{z,2} j) \in \mathcal{D}_s(x)\} \end{aligned} \tag{1}$$

2 LNS for Robust Solutions (LNSR)

LNSR takes an initial solution and gradually improves it by alternately destroying and repairing the solution using CP and a Branch and Bound algorithm [3, 5](Figure 1a). LNSR require a CSP solver modified in a way that it builts and maintains consistent the $\mathcal{N}_k$ sets ((Equation 1)) of the assigned variables for which robustness is required (not necessarily all the variables). The objective function is $f(s,k,\oplus) = \sum_{x \in \mathcal{X}_s} |\mathcal{N}_k(x, s(x), s, \oplus)|$ and the bound estimation for the robustness of the unassigned variables is $max(|\oplus| * k, domSize)$. LNSR uses a fixed neighbourhood size (20% of the variables) and it applies two neighborhood heuristics: random, when no better solution was found in the previous iteration, or *Wdeg/domSize* [2] heuristic in other case. *Wdeg/domSize* is always used as heuristic variable selection. We designed a *Robust* value selection heuristic that chooses the value that invalidates the less number of values in the $\mathcal{N}_k$ set of the assigned variables. As usual in CP, we use a failure limit ($fLim$) during the repairing process. After every LNSR iteration, $fLim$ is updated as $fLim = 100 * (1.1^{neighC})$ ($neighC$ is the number of neighborhoods explored).



(a) LNRS



(b) Taillard-js-020-15-9 (k=1)

3 Evaluation and Conclusions

We implemented LNSR and the Branch and Bound algorithm from [3] (denoted as B&B) into the ACE solver [4]. We evaluated both algorithms in Random CSPs [1] and Job Shop Scheduling Problems (JSP) [2] with a fixed makespan (Our proposal is only applicable to CSP for the moment). B&B-2' and LNSR-2 (the two best variants that we implemented, please review the complete version for more details) use *First* value selection until reaching the first solution. They also use *First* value in the subsequent restarts but switch to *Robust* when there are as many unassigned variables as the neighborhood size. Figure 1b shows the solutions found over time for B&B-2' and LNSR-2 for $k = 1$ in a particular scheduling instance. For other instances, executions and k's, the graphs are similar (even in general random CSPs). Note that LNSR-2 finds a much higher number of solutions than B&B-2'. The robustness of the solutions found by LNSR-2 increments quickly over time; however, B&B-2' finds solutions that have similar robustness between them. In addition, B&B-2' is unable to improve the robustness after a short time.

[1] We used the random generator downloable in https://www.lirmm.fr/ bessiere/generator.html. We modified it to produce XCSP3 output format.
[2] Downloable from http://www.xcsp.org/instances/

References

1 Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

2 Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. *Frontiers in Artificial Intelligence and Applications*, 110:146–150, 2004.

3 Laura Climent, Richard J. Wallace, Miguel A. Salido, and Federico Barber. Robustness and stability in constraint programming under dynamism and uncertainty. *Journal of Artificial Intelligence Research*, 49:49–78, 2014.

4 Christophe Lecoutre and Sebastien Tabary. Abscon 112 : towards more robustness. In *3rd International Constraint Solver Competition*, pages 41–48, 2013.

5 Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1520:417–431, 1998.