On the comparison of CMSA versus LNS for solving Combinatorial Optimization problems with different solution sizes

Evelia Lizárraga[1], Maria J. Blesa[1], Christian Blum[2]

[1] Computer Science Department, Universitat Politècnica de Catalunya
UPC North Campus, 08034 Barcelona
{evelial,mjblesa}@cs.upc.edu
Authors' work partially supported by AGAUR from the Generalitat de Catalunya (project SGR 2014-1034) and the Mexican National Council for Science and Technology (CONACYT, doctoral grant number 253787).

[2] Artificial Intelligence Research Institute, Spanish National Research Council (IIIA-CSIC)
UAB Campus, 08193 Bellaterra
christian.blum@iiia.csic.es

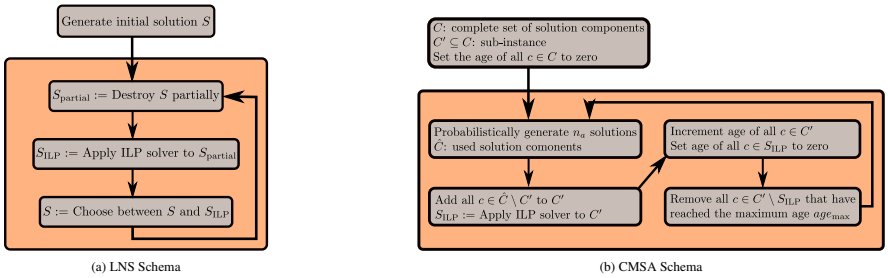
Abstract

Both, Construct, Merge Solve and Adapt (CMSA) and Large Neighborhood Search (LNS), are hybrid algorithms that are based on iteratively solving sub-instances of the original problem instances, if possible, to optimality. This is done by reducing the search space of the tackled problem instance in algorithm-specific ways which differ from one technique to the other. In this paper we provide first experimental evidence for the intuition that, conditioned by the way in which the search space is reduced, LNS should generally work better than CMSA in the context of problems in which solutions are rather large, and the opposite is the case for problems in which solutions are rather small. The size of a solution is hereby measured by the number of components of which the solution is composed, in comparison to the total number of solution components. In this ongoing work we are conducting experiments in the context of the multi-dimensional knapsack problem, the minimum-weight dominating set, and the single-source capacitated facility location problem.

1 Introduction

Construct, Merge Solve and Adapt (CMSA) [3] and Large Neighborhood Search (LNS) [7] are general purpose algorithms for solving combinatorial optimization problems. Both techniques are hybrid approaches that combine the application of a general ILP solver (or any other exact method) to reduced problem instances. In this context, the terms *reduced problem instance* and *sub-instance* refer to a subset of the set of solutions to the tackled problem instance which are obtained by search space reduction. The general idea for both algorithms is the following one: to identify a substantially reduced sub-instance of a given problem instance such that the sub-instance contains high-quality solutions to the original problem instance. This might allow one to apply an exact technique—such as, for example, an ILP solver—with reasonable computational effort to the reduced sub-instance in order to obtain a high-quality solution to the original problem instance. In other words, both algorithms employ techniques for reducing the search space of the tackled problem instances. The general purpose ILP solver is then used to find the best solution in the reduced search space.

Although both approaches are based on the same general idea, the way in which the search is reduced differs from one to the other. LNS keeps an incumbent solution which, at each iteration, is partially destroyed. This results in a partial solution. The reduced search space consists of all solutions to the original problem instance that contain this partial solution. CMSA, on the other side, reduces the search space as follows: at each iteration, solutions to the original problem instance are constructed in a probabilistic way, using a greedy function as bias. The *solution components* found in these solutions are joined together, forming a subset C' of the complete set of solution components. The set of solutions to the original problem instance that can be generated on the basis the components in C' form the reduced search space in CMSA.



(a) LNS Schema (b) CMSA Schema

Figure 1: LNS and CMSA in brief.

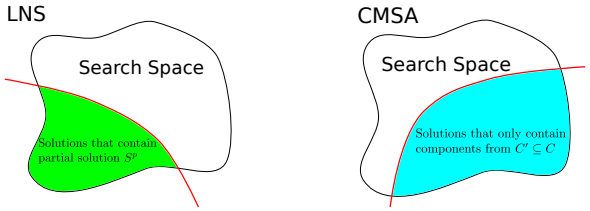


Figure 2: Search space reduction in LNS and CMSA.

2 Our current ongoing work

Although both LNS and CMSA are based on the same general idea, the way in which the search space is reduced differs from one to the other. Based on this difference we had the intuition that LNS would (generally) work better than CMSA for problems for which solutions are rather large, and the opposite would be the case in the context of problems for which solutions are rather small. The size of solutions is hereby measured by the number of solution components (in comparison to the total number) of which they are composed. For example, in the case of the travelling salesman problem, the complete set of solution components is composed of the edges of the input graph. Moreover, solutions consist of exactly n components, where n is the number of vertices of the input graph. The above-mentioned intuition is based on the consideration that, for ending up in some high-quality solution, LNS needs to find a path of over-lapping solutions from the starting solution to the mentioned high-quality solution. The smaller the solutions are, the more difficult it should be to find such a path.

A theoretical validation of our intuition seems, a priori, rather difficult to achieve. Therefore, we decided to study empirical evidence that would support (or refute) our intuition. For this purpose, we considered three widely studied combinatorial optimization problems as case studies: the *multi-dimensional Knapsack problem* (MDKP), the *minimum weight dominating set* problem (MWDSP), and the *single-source capacitated facility location problem* (SSCFLP). For these NP-hard problems, it is possible to generate both, problem instances for which solutions are small and problem instances for which solutions are large. We implemented both LNS and CMSA for all these problems and performed an empirical study of the results of both algorithms for problem instances over the whole range. It should be noted that LNS has been previously applied to the MWDSP [4] and CMSA to the MDKP [2] separately. Both with considerable success.

In the context of the MDKP problem, our work in [5] has shown the first empirical evidence that supports our initial intuition that LNS should generally work better than CMSA for problems in which solutions contain rather many solution components, and vice versa. We are currently performing a deep experimental work for the MWDSP problem and the SSCFLP problem, where a similar behavior seems to happen.

In the near future we intent to confirm this empirical evidence by the application to additional optimization problems. From a theoretical point of view, we would need to develop the tools for characterizing which problems are likely to be solved by LNS and which by CMSA. But that is another tale for another time.

3 Notes

Finally, we would like to clarify the following aspect. Our intuition obviously only holds for problems for which, a priori, neither LNS nor CMSA have advantages over the other one. In fact, it is not very difficult to find problems for which CMSA generally has advantages over LNS, no matter if solutions are small or large. Consider, for example, problems for which the number of variables and/or constraints in the respective ILP model are so large that the problem cannot be solved simply because of memory restrictions. This is the case in ILP models in which the number of variables and/or constraints are superlinear concerning the input parameters of the problem. Due to its specific way of reducing the search space, CMSA tackles sub-instances that correspond to reduced ILP models. This is not the case of LNS. Even though parts of the solution are fixed, the complete original ILP model must be built in order to solve the corresponding sub-instance. Therefore, CMSA can be applied in these cases, while LNS cannot be applied. An example of such a problem is the repetition-free longest common subsequence problem [1]. Contrarily, it is neither difficult to think about problems for which LNS generally has advantages over CMSA. Consider, for example, a problem where the main difficulty is not the size of ILP model but rather the computational complexity. Moreover, let us assume that when fixing a part of the solution, the sub-instance becomes rather easy to be solved, which is—for example—the case in problems with strong symmetries. In such a case LNS will most probably have advantages over CMSA. An example of such a problem is the most strings with few bad columns problem [6].

References

[1] C. Blum and M. Blesa. Construct, merge, solve and adapt: Application to the repetition-free longest common subsequence problem. In *16th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2016)*, number 9595 in LNCS series, pages 46–57. Springer, 2016.

[2] C. Blum and J. Pereira. Extension of the CMSA algorithm: An LP-based way for reducing subinstances. In *Genetic and Evolutionary Computation Conference (GECCO 2016)*, pages 285–292, New York, NY, USA, 2016. ACM.

[3] C. Blum, P. Pinacho, M. López-Ibáñez, and J.A. Lozano. Construct, Merge, Solve & Adapt: A new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68:75–88, 2016.

[4] C. Blum and G. R. Raidl. *Hybrid metaheuristics*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2016.

[5] E. Lizárraga, M. Blesa, and C. Blum. Construct, Merge, Solve and Adapt versus Large Neighborhood Search for Solving the Multi-Dimensional Knapsack Problem: Which One Works Better When? In *17th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2017)*, number 10197 in LNCS series. Springer, 2017. To appear. DOI: 10.1007/978-3-319-55453-2.

[6] E. Lizárraga, M. Blesa, C. Blum, and G. Raidl. Large neighborhood search for the most strings with few bad columns problem. *Soft Computing*, 2016. In press. DOI:10.1007/s00500-016-2379-4.

[7] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *4th International Conference on Principles and Practice of Constraints Programming (CP-98)*, volume 1520 of *LNCS Series*, pages 417–431. Springer, 1998.