An analysis on the destroy and repair process in large neighbourhood search applied on the vehicle routing problem with time windows

J. Corstjens, A. Caris, and B. Depaire
UHasselt, Research Group Logistics
e-mail: jeroen.corstjens@uhasselt.be

For many years experimenters have successfully resorted to heuristic algorithms when solving complex optimisation problems, ranging from construction methods to high-level metaheuristic frameworks. Whenever such a method is presented, the added value it delivers is shown in a competitive evaluation context. This means that the algorithm is applied to solve the instances of one or several well-known benchmark problem sets after which its performance on these instances is compared to the performance of other algorithms that solved the same instances. The aim is to show that the presented algorithm performs better, in the sense of a better solution quality, computation time or trade-off of both performance measures. One rarely sees a thorough investigation of how an algorithm establishes its performance. What elements of the algorithm contribute the most to performance? How does this contribution depend on the problem characteristics? What can we learn about any performance difference observed between distinct parameter configurations? Such insights are necessary to truly understand algorithmic behaviour [5].

In recent years, there has been increased focus on identifying the algorithm elements that are most relevant to performance ([2, 3, 4]). We want to go a step further and understand why these elements work well or not, enabling us to explain why it is that two parameter configurations or two distinct algorithms differ in the performance they obtain. That is the type of question we aim to address when experimenting with heuristic algorithms. In a case study we analyse a large neighbourhood search (LNS) algorithm applied on instances of the vehicle routing problem with time windows (VRPTW). A first exploratory analysis exposed several patterns raising questions that are to be answered in new consecutive experiments [1]. One observed pattern concerns certain combinations of destroy and repair operators. More specifically, a first experimental analysis showed that removing customers at random from a solution each iteration leads to a better performance than removing geographical clusters of customers if these customers are to be reinserted using a regret measure that takes into account which customers are more difficult to insert and should be prioritised. In this follow-up research we wish to understand why there is a performance difference.

Searching for explanations, the focus shifts from a complete large neighbour-

hood search framework to a single destroy and repair iteration. A data set of 10,000 single iteration observations is generated, consisting of 200 VRPTW instances and 50 parameter settings per problem instance. A parameter setting is interpreted in this experiment as a combination of a single destroy operator with a single repair operator. Solutions are destroyed using either random or related removal, while they are repaired using a regret-k operator with k equal to 2,3 or 4.

The observed performance difference between the destroy operators in the LNS experiment is also observed in the new experiment. Given this confirmation, the search for explanations started by decomposing the destroy and repair process. The analysis showed that the removal of a cluster of customers that are geographically nearby is detrimental for the repair phase as it reduces the number of alternative routes available. The latter is crucial for a regret operator if it wants to make the best insert decisions. Even more, some customer no longer have any feasible alternative in one of the existing routes and are considered to be isolated cases. It is found that prioritising these customers by inserting them in individual routes reduces the majority of the performance gap between random and related removal. The creation of individual routes early on in the repair phase benefits all customers that are to be inserted as it increases the number of available alternatives. Hence, a regret operator will make a better estimation of customer difficulty and consequently a better prioritisation if each individual customer has existing routes nearby in which it can be feasibly inserted.

We still observe a small significant performance difference between random and related removal, but before looking for further explanations we will perform an experiment on a complete large neighbourhood search algorithm. In this experiment it will be tested whether the findings for the one iteration experiment also hold when performing a normal LNS experiment that runs many more iterations.

References

[1] Corstjens, J., Depaire, B., Caris, A., & Sörensen, K. (2016). Analysing metaheuristic algorithms for the vehicle routing problem with time windows. In Verolog 2016 proceedings, 89.

[2] Fawcett, C., & Hoos, H. (2015). Analysing differences between algorithm configurations through ablation. Journal of Heuristics, 22(4), 431-458.

[3] Hutter, F., Hoos, H., & Leyton-Brown, K. (2015). Identifying key algorithm parameters and instance features using forward selection. Lecture Notes in Computer Science, 7997, 364-381.

[4] Hutter, F., Hoos, H., & Leyton-Brown, K. (2015). An efficient approach for assessing hyperparameter importance. International Conference on Machine Learning, 754-762.

[5] Rardin, R. L., & Uzsoy, R. (2001). Experimental evaluation of heuristic optimization algorithms: A tutorial. Journal of Heuristics, 7(3), 261-304.