Explaining metaheuristic performance through iterative experimentation

Explaining metaheuristic performance through iterative experimentation

Jeroen Corstjens[1], An Caris[1] and Benoît Depaire[2]
[1]UHasselt, Research Group Logistics, Belgium, jeroen.corstjens@uhasselt.be
[2]UHasselt, Research Group Business Informatics, Belgium

Abstract
For many years metaheuristics have been successfully applied to solve computationally challenging optimisation problems. These general solutions procedures are most commonly evaluated by running them on standard benchmark problems and comparing performance results with other state-of-the-art methods. The objective is to be better than the competition. A detailed investigation of the metaheuristic elements responsible for the superior performance is rarely performed. Understanding how all the elements impact performance and how they interact with the specific problem instance to be solved is, nevertheless, relevant to gain insight into both metaheuristic and optimisation problem. In this research, the focus is on gaining a better understanding of heuristic algorithm performance. We investigate the performance difference between two configurations of a large neighbourhood search algorithm applied on instances of the vehicle routing problem with time windows and are able to substantially reduce the performance gap after a detailed analysis of the destroy and repair process. We observed that when reinserting customers in the solution, the most isolated ones should be prioritised.

Keywords: Experimental analysis, Metaheuristics, Understanding, Vehicle routing, Large neighbourhood search.

1 Introduction

Metaheuristics are widely employed to solve computationally challenging optimisation problems. They are usually applied in a competitive context: solving standard benchmark instances and then comparing performance results with those of state-of-art methods. The goal is to obtain better results than the competition. Investigating which algorithm elements actually contribute to the superior performance is not usually a research goal. Yet, experimenters control a lot of these elements as metaheuristics can be seen as consisting of several interacting heuristic mechanisms. Which of these mechanisms are activated, how they behave and interact with other mechanism can often be controlled by parameters. The choices made for these parameters substantially impact the efficacy with which a heuristic algorithm solves a given problem instance or class of problem instances [1].

Given the impact parameters have on performance, several procedures have been developed that automate the task of tuning these parameters [2]. They do not generally provide information on why one parameter choice works better than another one. What is the contribution of each element to performance? How does the contribution vary when considering different problem instances? Why does a (range of) value(s) work better than other values? Such questions are not commonly addressed when discussing experimental results of heuristic algorithms [3]. The attention usually focuses on achieving better performance results than competing algorithms [4]. However, this approach does not provide much insight into heuristic algorithm behaviour [5]. A competitive evaluation methodology is useful when the objective is to develop the fastest possible procedure for a specific environment. When the goal is to understand how performance is obtained, to discover which elements in the heuristic algorithm contribute to its performance and to draw conclusions that are valid beyond the specific problem instances chosen, one has to rely on a statistical evaluation method [6].

This research focuses on explaining why two configurations of a metaheuristic algorithm perform differently. What can be learned about the differences in components, strategies or implementations that result in an either better or worse performing heuristic method. Based on an exploratory analysis of the parameters' impact on performance, explanations are sought for the observed patterns through iterative experimentation. We observe data, ask questions about it, formulate possible answers and verify the answers in new experiments.

2 Ask a question

An analysis of a large neighbourhood search (LNS) algorithm solving a number of instances for the vehicle routing problem with time windows (VRPTW) is performed. The importance of VRPTW in many distribution systems has spurred intensive research efforts for both heuristic and exact optimisation approaches. Yet, few

research articles apply statistical techniques to evaluate heuristics for these problems or seek to understand how they influence heuristic algorithm behaviour. LNS is a popular metaheuristic framework applied to multiple VRP variants in which an initial solution is gradually improved by iteratively removing and reinserting customers in a solution until some stopping criterion is met. This iterative process uses a destroy and repair operator.

The exploratory analysis in [7] is performed on a data set of 4000 observations. The data set is generated in two phases. Problem instances are first sampled and then, per generated instance, a number of parameter settings are randomly defined. Multilevel regression models are formulated since this type of data structure violates the assumption of independent error terms made by traditional regression analysis. The results exposed correlations between algorithm parameters and performance and between algorithm parameters and problem instance characteristics. One notable observation is that iteratively removing customers at random from a solution is expected to result in a better performance than iteratively removing clusters of customers when in both cases these customers are reinserted based on a regret measure that prioritises difficult customers. Since it seems counterintuitive for a destroy operator relying purely on randomisation (i.e., random removal) to perform better than an operator employing a more sophisticated logic (i.e., related removal) given their combination with a certain repair operator (i.e., regret-2), the question is raised what is special about these specific combinations that leads to the observed result. Therefore, we now focus on these operators and try to explain the performance difference.

3 Search for answers

Since the focus of this follow-up research is on analysing individual operators and not a complete metaheuristic, the operators are extracted from the metaheuristic framework and the solution quality after a single destroy and repair iteration is measured. This will allow detailed analysis of the destroy and repair process.

First, the selection criteria either to remove or insert customers are examined. Random removal randomly selects q customers to remove. The idea is to diversify the search towards a better solution. Related removal defines a relatedness criterion to base removals on. We define relatedness in terms of distance as in [8], looking at the geographical closeness of two customers. The idea is that customers close to each other can more easily switch routes and/or positions, while more distanced customers have a higher chance of being inserted back in their original position. In both removal cases, customers are reinserted using a regret-k operator with k equal to 2, 3 or 4. This operator prioritises customers that are considered 'difficult'. The difficulty criterion used to decide which customer to iteratively insert is the additional cost incurred when not inserting a customer in its best route. The higher this additional cost, the higher the priority given to the customer. This measure is also referred to as the regret value. It is calculated as the sum of the difference between a customer's cheapest insertion route (i.e., insertion in this route adds the smallest distance to the overall travelled distance) and its second, third, … up until its k-th cheapest insertion route. Hence, customers with large regret values have large cost gaps between their best and second, third, fourth, … best insertion route. Therefore, these customers should be considered first for insertion, since they only have a small number of interesting insertion alternatives. Customers having small regret values do not have to be immediately inserted since they can more easily be inserted in alternative routes for which the cost of insertion is not a lot higher compared to the best insertion route. Since it is the key measure used in the repair process and applied for both removal scenarios, a first analysis focused on this measure and how it differs for inserting randomly dispersed customers and for a clustered group of customers.

We observed that random removal leads to solutions with similar structure to the initial solution. During the repair process only small adjustments are made that potentially result in a better final solution. When removing a geographical cluster of customers, on the other hand, a (large) part of the solution structure is completely destroyed and has to be rebuilt from scratch (cf. figure 1). For many of the removed customers, there are not a lot of existing routes nearby that are potential candidates to be inserted in. Consequently, the number of good alternatives for their cheapest insertion route is expected to be small. A randomly removed customer, however, has more existing routes nearby and thus better alternatives for the cheapest insertion route, resulting in a lower regret value. The expectation therefore is that a removed customer who is part of a geographical cluster of removed customers has less feasible alternatives compared to a randomly removed customer. Further, due to the difference in feasible alternatives, it is expected that the average regret value of the selected customer for insertion is higher for scenarios in which a cluster of customers has been removed. This is formulated in hypotheses 1 and 2.

(a) Random removal (b) Related removal

Figure 1: Destroyed solution for a problem instance with 100 customers

Hypothesis 1 (H1): *When a cluster of geographically nearby customers is removed, each removed customer has on average the same number of feasible routes to be inserted compared to a customer that was removed at random.*

Hypothesis 2 (H2): *The average regret value of the selected customer for insertion per iteration does not differ when customers are removed at random compared to when a cluster of geographically nearby customers is removed.*

Having less feasible options and a higher difficulty measure, what does this imply for the way customers are reinserted in the solution? We notice that the majority of the customers that are inserted first are almost all in near proximity of an existing route, while the more "isolated" customers are all postponed to the final insertions. What characterises these "isolated" customers is that they cannot be feasibly inserted in one of the existing routes. This means that their regret value is zero and they are thus assigned the lowest priority of all removed customers. The single alternative they have is to create a route straight from the depot to the customer and back. Such an alternative is assigned a regret value of zero since there is no cost loss of postponing the insertion for that customer. The customer has no other alternatives, so its insertion is not considered urgent. In addition, as the solution of routing problems typically not only strives to minimise the total distance travelled, but also doing so with the fewest number of vehicles necessary, the current priority mechanism does not favour the creation of additional routes. Nonetheless, this observation raises the question what the impact would be if these isolated customers are taken into account at the start of the repair process instead of initially being ignored. Perhaps their prioritisation might benefit other removed customers as this adds alternative routes in an area with few or no routes at all, perhaps better alternatives than when the isolated customers are postponed to the final insertions. Therefore, we hypothesise that removing a cluster of customers results in more customers having no other feasible insertion option than an individual route from the depot to the customer and back than when removing customers at random. Secondly, we hypothesise the solution quality for scenarios using related removal to be better when prioritising isolated customers first instead of postponing their insertion to the final ones.

Hypothesis 3 (H3): *The number of removed customers whose sole feasible option at the start of the repair process is a route from the depot to the customer and back is the same when the removed customers are chosen randomly or when they belong to a cluster of customers.*

Hypothesis 4 (H4): *When a cluster of geographically nearby customers is removed, prioritisation of the customers with no other feasible insertion option than an individual route from the depot and back does not result in a better solution quality than when these customers are postponed to the final insertions.*

4 Validate hypotheses

The hypoteses are validated in a new experiment. A data set of 10,000 observations is generated following the same multilevel experimental design as employed in [7]. It consists of 200 artificial VRPTW instances and 50

random parameter settings tested per problem instance. A parameter setting is defined as a combination of a single destroy operator — either random or related removal — with a single repair operator — either regret-2, regret-3 or regret-4.

The performance data is analysed relying on multilevel regression analyses. All hypotheses are validated in separate regression analyses using the *brms* package in R. For hypotheses 1, 2 and 3 statistical evidence is found, with at least 95% confidence, to reject them. In other words, scenarios relying on related removal have significantly less feasible insert options for a removed customer, have on average a higher regret value than scenarios relying on random removal and have a significantly higher number of removed customers with no other feasible insertion option than an individual route from the depot and back.

For the validation of the fourth hypothesis the priority mechanism is modified such that the most isolated customers are now prioritised at the start instead of being ignored. The 10,000 scenarios are run again using the adjusted prioritisation. The regression output showed an overall better solution quality and, more importantly, a substantial reduction in the performance difference between random and related removal. For problem sizes of about 200 customers or more, related removal still performs significantly worse than random removal.

5 Conclusion

We are able to explain (part of) the observed performance difference between two configurations of the same metaheuristic algorithm. Analysis of a single destroy and repair iteration showed that removing a geographical cluster of customers results in fewer interesting alternative routes compared to removing customers at random. These customers are therefore assigned a higher difficulty. In such a scenario it is important to prioritise those customers who can only be inserted in a route straight from the depot to the customer and back. This results in a smaller performance gap between scenarios using related removal and scenarios using random removal.

The experimental data generated to validate the hypotheses is based on a single destroy and repair iteration performed on an initial solution. Whether our findings hold when running multiple iterations as in the large neighourhood search framework is to be verified in a experiment similar to the one performed by [7] using new sample data. We await these analysis results before looking to explain the remaining significant performance difference. One possible idea to look into is which isolated customers to prioritise first.

References

[1] H. Hoos. Automated Algorithm Configuration and Parameter Tuning. *Autonomous Search*, 37–71, 2011.

[2] M. Birattari, and J. Kacprzyk. Tuning metaheuristics: a machine learning perspective *Studies in Computational Intelligence*, 197, 2009.

[3] C. Fawcett, and H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458, 2015.

[4] K. Smith-Miles, and S. Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, 2015.

[5] T. Bartz-Beielstein, and M. Preuss. The future of experimental research. *Experimental Methods for the Analysis of Optimization Algorithms*, 17–49, 2010.

[6] R. Rardin, and R. Uzsoy. Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. *Journal of Heuristics*, 7(3):261–304, 2001.

[7] J. Corstjens, B. Depaire, A. Caris, and K. Sörensen. A Multilevel Evaluation Method for Heuristics with an Application to the VRPTW. *Manuscript sumbitted for publication*, 2017.

[8] D. Pisinger, and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.