Decomposition strategies for vehicle routing heuristics

Alberto Santini[1], Michael Schneider[2], Thibaut Vidal[3], and Daniele Vigo[4]

[1]Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, Spain*
[2]Deutsche Post Chair - Optimization of Distribution Networks, RWTH Aachen University, Germany†
[3]Department of Computer Science, Pontifical Catholic University of Rio de Janeiro, Brazil‡
[4]DEI and CIRI-ICT, Alma Mater University of Bologna, Italy§

January 8, 2023

Abstract

Decomposition techniques are an important component of modern heuristics for large instances of vehicle routing problems. The current literature lacks a characterisation of decomposition strategies and a systematic investigation of their impact when integrated into state-of-the-art heuristics. This paper fills this gap: we discuss the main characteristics of decomposition techniques in vehicle routing heuristics, highlight their strengths and weaknesses, and derive a set of desirable properties. Through an extensive numerical campaign, we investigate the impact of decompositions within two algorithms for the capacitated vehicle routing problem: the Adaptive Large Neighbourhood Search of Pisinger and Ropke (2007) and the Hybrid Genetic Search of Vidal et al. (2012). We evaluate the quality of popular decomposition techniques from the literature and also propose new strategies. We find that route-based decomposition methods, which define subproblems by means of the customers contained in selected subsets of the routes of a given solution, generally appear superior to path-based methods, which merge groups of customers to obtain smaller subproblems. The newly proposed decomposition *barycentre clustering* achieves the overall best performance and leads to significant gains compared to using the algorithms without decomposition.

Keywords: vehicle routing; heuristics; decomposition methods

1 Introduction

Vehicle Routing Problems (VRPs) call for the determination of minimum-cost vehicle routes to serve a set of geographically dispersed clients. Due to their practical relevance for distribution logistics and their notorious difficulty, VRPs have been the focus of extensive research, counting hundreds of papers proposing exact and heuristic solution methods (see, e.g., Toth and Vigo 2014; Vidal et al. 2020). Despite significant advances in exact solution approaches in recent years (see, e.g., Pecin et al. 2017; Costa et al. 2019; Pessoa et al. 2019), heuristics still remain indispensable for VRP variants with complicating attributes (additional decisions, constraints and objectives) arising from practical applications and for solving large-scale instances (Vidal et al. 2013b).

In the heuristic domain, decomposition techniques have proven their worth in solving large (see, e.g., Groër et al. 2011; Vidal et al. 2014; Uchoa et al. 2017) or very large VRP instances (see, e.g., Arnold et al. 2019), and they are also successfully used to boost the performance of heuristics on medium-sized instances (Goeke et al. 2019). Decomposition strategies are widely used in practice because they generally lead to more structured and intuitive routing plans, e.g., by assigning clusters of geographically close customers to the same route (Rossit et al. 2019, Sec. 2.1). However, we are not aware of a systematic

*alberto.santini@upf.edu
†schneider@dpo.rwth-aachen.de
‡vidalt@inf.puc-rio.br
§daniele.vigo@unibo.it

characterization and investigation of the performance of different decomposition techniques for VRPs. In this paper, we aim to fill this gap with a twofold contribution:

1. We discuss some fundamental characteristics of heuristic decomposition techniques for VRPs and link these characteristics to recent papers on this topic. We adopt a broader perspective on decomposition techniques and include:
   (a) Methods that repeatedly decompose the instances into smaller subproblems solved separately, and merge the solutions of the subproblems to obtain a complete solution of the original problem (see, e.g., Bent and Van Hentenryck 2010; Groër et al. 2011; Vidal et al. 2013a).
   (b) *Coarsening and aggregation* methods that fix arcs to join together nodes (see, e.g., Walshaw 2002; Santini 2019; Rodrigues de Holanda Maia et al. 2020), resulting in an effective decrease of the number of customers considered in the problem.
   (c) *Ruin-and-recreate* methods (see, e.g., Shaw 1998; Schrimpf et al. 2000; Ropke and Pisinger 2006b; Ropke and Pisinger 2006a), which temporarily keep a set of customers and arcs as fixed, and attempt to rearrange the rest of the decision variables.
2. We experimentally investigate the impact of different decomposition techniques within state-of-the-art metaheuristics for the Capacitated VRP (CVRP) to derive methodological guidelines for future applications. In particular, we evaluate the performance of two popular high-quality algorithms when using different decomposition techniques: the Adaptive Large Neighborhood Search (ALNS) of Pisinger and Ropke 2007 and the Hybrid Genetic Search (HGS) of Vidal 2022. In our experiments, we focus on *homogeneous* decompositions (see Section 2.1), that create subproblems of the same nature as the original problem and permit to use the same solver for the main problem and the subproblems. We rely on the large-scale instances of Uchoa et al. 2017, which possess diverse characteristics (e.g., customer distribution, route length, depot positioning, demand patterns) and still represent a challenge for modern metaheuristics.

We run our experiments on the CVRP because it is the canonical variant of the VRP family. The CVRP can be formally defined on a complete undirected graph $G = (V, E)$, where $V = \{0\} \cup V'$ is the set of locations, containing a depot node 0 and a set of customers $V' = \{1, \ldots, n\}$, and E is the set of all edges between locations. Each customer $v \in V'$ has a demand of $q_v \geq 0$ units, and a non-negative service time. Each edge $(u, v) \in E$ represents the possibility of traveling between locations u and v at a cost c_{uv}. In the context of this work, we also assume that the geographical coordinates (x_u, y_u) of each location $u \in V$ are known. A fleet of p identical vehicles with capacity Q is available to serve the customers. The goal of the CVRP is to determine up to p routes, each route starting at the depot, visiting a sequence of customers and returning to the depot, in such a way that the total customer demand on each route does not exceed the vehicle's capacity, that each customer is visited once, and that the total travel cost is minimized.

The remainder of this paper is structured as follows. In Section 2, we discuss the main characteristics of heuristic decomposition techniques and illustrate this characterization on a number of recent works. In Section 3, we give a detailed description of the decomposition methods considered in our numerical analyses. In Section 4, we present our experimental result and discuss a number of design recommendations. Finally, in Section 5, we conclude the paper and highlight future research directions.

2 Characteristics of decomposition techniques

Heuristic decomposition strategies stem from the divide-and-conquer principle. They define one or several subproblems in such a way that their solution contributes to the solution of a complex original problem, either by producing a new solution, improving an existing one, or by identifying promising or unpromising regions of the search space. Decomposition techniques are intimately connected to *projection* strategies (see, e.g., Geoffrion 1970a; Geoffrion 1970b), which involve fixing some of the decision variables to focus the search in a smaller subspace. With this view in mind, heuristic decomposition techniques may be characterized in relation to:

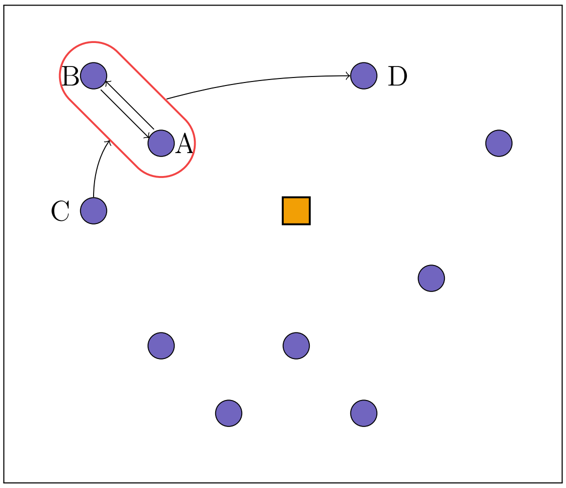


Figure 1: Illustration of an heterogeneous decomposition: Aggregating nodes A and B does not permit to define distances towards the resulting macro-node AB if the internal traversal direction is undefined.

1. the nature of the subproblems formed by the decomposition;
2. the information used to define the subproblems;
3. the solution methods for the subproblems;
4. the ways the subproblem solutions are used to solve the master problem.

We will now discuss these aspects in more detail and position recent works in relation to them.

2.1 Nature of the subproblems

Homogeneous or heterogeneous decomposition. A decomposition is *homogeneous* if it leads to subproblems that retain the same definition and structure as the original problem. Such a decomposition permits to use a single solution approach recursively. In contrast, an *heterogeneous* decomposition may require tailored techniques for the solution of the subproblems. Decompositions based on customer partitions are usually homogeneous and lead to vehicle routing problems of a smaller scale. In contrast, some decompositions that aggregate successive customers into bigger nodes (Walshaw 2002; Chevalier and Safro 2009) are heterogeneous because the "macro-nodes" representing these visit sequences in the subproblem may be visited in two possible directions (in a similar fashion as services on edges in arc-routing problems (Vidal 2017)). Figure 1 illustrates this issue when merging two nodes A and B into a macro-node AB. In the subproblem resulting from the aggregation, a vehicle may visit node C, followed by AB and then by D. However, depending on the positions of C and D, it could be more advantageous to visit A before B, or B before A, making it impossible to define a single cost matrix connecting to and from AB in the subproblem.

Independent or dependent subproblems. Many decompositions form multiple subproblems at the same time. The subproblems are *independent* when the objective value and feasibility of one subproblem does not depend on the others. Most decomposition techniques based on customer partitions (e.g., Taillard 1993; Vidal et al. 2013a; Goeke et al. 2019), called partitional decompositions in this paper, induce independent subproblems with disjoint subsets of customers.

There exist some exceptions though. For example, Groër et al. 2011 assign all customers of a route to one of two subproblems if the route has at least one customer lying in the corresponding half-plane. Customers located in routes at the intersection of the half-planes appear in both subproblems. In the presence of side constraints, partitional decomposition may no longer form independent subproblems. Examples are workload balancing between routes, levels of service for some subsets of customers (Bulhões et al. 2018), or consistency constraints (Groër et al. 2009). Finally, even a global fleet-size limit may introduce some dependencies between the subproblems. To avoid this dependency, one can define a maximum number of vehicles for each subproblem in a manner that is consistent with the global limit.

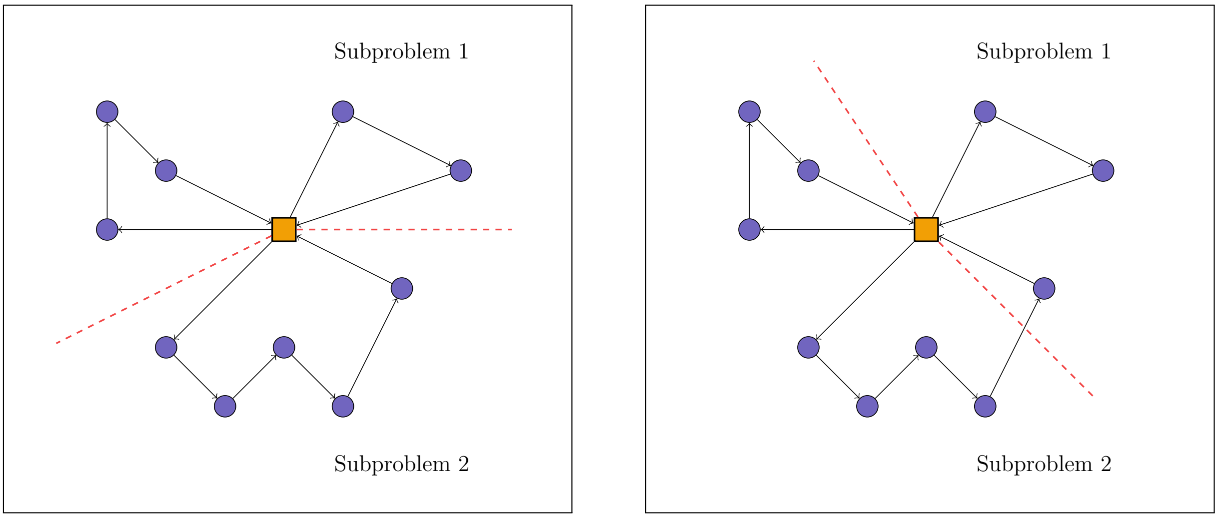


Figure 2: Decomposition supported (left) and non-supported (right) in an elite solution, whose routes are indicated by arrows. On the left, the customers visited by the same vehicle are all assigned to the same subproblem. On the right, there are customers which are visited by the same vehicle (in the bottom route) but are assigned to different subproblems.

Another class of decomposition methods which produces dependent subproblems are ruin-and-recreate algorithms (Shaw 1998; Schrimpf et al. 2000; Ropke and Pisinger 2006b; Ropke and Pisinger 2006a). These algorithms improve the solutions by iteratively ruining a part of the solution while the rest remains fixed. This is equivalent to a decomposition approach which iteratively generates and solves smaller subproblems. Because there can be overlap between the ruined parts from one iteration to the next, the solution of these subproblems is typically done sequentially.

Overall, independent decompositions are usually desirable because they allow high-level parallelism to reduce wall-clock time and permit one to combine the solutions of the subproblems (see Section 2.4) in a straightforward way.

2.2 Information used to define the subproblems

Decomposition approaches are usually designed to obey three main principles:

1. Variable fixings induced by the decomposition (as seen in terms of fixed or prohibited arcs) should be supported in high-quality solutions;
2. The free decision variables of the subproblem should not be too few or too numerous in such a way that the subproblems are non-trivial but easier than the original problem;
3. The free decision variables should be related, so that their solution contributes to the search.

Different strategies have been developed to reach these goals, as discussed in the following.

Support in an elite solution. It is common to rely on the characteristics of a high-quality (i.e., *elite*) solution to define the decomposition. For example, forming subproblems from the customers of a subset of routes selected from an elite solution guarantees that the routes that are not currently selected appear together in at least one good solution. This effectively intensifies the search effort around this solution. Using the information of a single elite solution prevents a classical pitfall of some decomposition techniques: inducing solution features (e.g., arcs) which are frequently present in good solutions but never found together (supported) in at least one good solution. Figure 2 shows examples of decompositions respectively supported and *not* supported in an elite solution.

Methods based on partitioning the set of customers are no exception to this concept. Splitting the set of customers to create two CVRP subproblems implicitly means that if two customers belong to different subproblems, then no arc can exist between them in the associated complete solution. Therefore, as a prerequisite for such a partition, one should be sure that at least one good solution exists without this arc. A similar issue occurs when decomposing a problem by partitioning the customers into subsets $V_1, \ldots, V_k$ such that the minimum number of routes required in the solutions of the subproblems, $\sum_{j=1}^{k} \lceil (\sum_{v \in V_j} q_v)/Q \rceil$, is much greater than the minimum number of routes required in a solution of the original problem, $\lceil (\sum_{v \in V'} q_v)/Q \rceil$. High-quality complete solutions are unlikely to have such a structure because the total residual space in the vehicles is generally small. This can be avoided if the partition is based on routes from an elite solution. Alternatively, pattern mining techniques (such as frequent item sets detection) have been applied to detect groups of edges that jointly appear in high-quality solutions. Having detected those edges, it is then possible to contract them and aggregate the associated nodes to create subproblems (see, e.g., Ribeiro et al. 2006; Rodrigues de Holanda Maia et al. 2020). Another related approach consists in reintroducing frequent groups of edges (patterns) jointly, through dedicated local-search moves (Arnold et al. 2021).

Size of the subproblems. The size of the subproblems is an essential parameter to control their difficulty and potential for improvement. Good choices of problem size depend on the performance (run-time and solution quality) of the algorithm adopted for the subproblems. The aim is to choose a problem size such that the solver is able to solve the subproblems to near-optimality in a limited time. The size of the subproblems can stay fixed, be subject to randomization (see, e.g., Ropke and Pisinger 2006a), or evolve during the search in relation to some performance metrics (see, e.g., Queiroga et al. 2020). Subproblems of fixed size are adequate in situations in which one has a precise knowledge of the capabilities of the subproblem solver. Adaptive schemes are useful if this information is not available or if small differences of instance characteristics can have a large impact on the performance of the solver. For example, when using an optimal algorithm to solve the subproblems, a few dozen customers or some structural differences in the customers' distribution can make the difference between solving the subproblems to optimality or not. In this case, Queiroga et al. 2020 adopt an iterative approach in which the subproblem size is progressively increased.

Relatedness information. Subproblems should not be trivial. Besides an adequate choice of problem size, creating non-trivial problems usually requires selecting customers and decision variables that are related. *Relatedness* between decision variables or customers can be measured in different ways:

- Spatial relatedness (Shaw 1997; Taillard 1993; Ropke and Pisinger 2006a; Groër et al. 2011; Vidal et al. 2013a);
- Temporal relatedness, in case of time-window-constrained problems (Bent and Van Hentenryck 2010; Shaw 1997; Ropke and Pisinger 2006a);
- Historical relatedness (Ropke and Pisinger 2006a; Ropke and Pisinger 2006b);
- Relatedness as observed from recent solution patterns (Arnold et al. 2021; Rodrigues de Holanda Maia et al. 2020);

Spatial and temporal relatedness metrics directly derive from the characteristics of the instances and measure how close in space or time two customer visits are. In contrast, historical and pattern relatedness examine how often certain decisions are taken together in good solutions in the search history. Note that some decomposition approaches rely on multiple relatedness metrics, possibly in an adaptive fashion (Ropke and Pisinger 2006a), and that linear combinations of these metrics can also be exploited (Shaw 1998; Schrimpf et al. 2000).

2.3 Solution techniques for the subproblems

Solution techniques for the subproblems can widely vary in terms of their run-time and solution quality. They range from simple greedy reconstruction heuristics (as is the case for most ruin-and-recreate applications) to more sophisticated metaheuristics applied recursively on the subproblems (Groër et al. 2011; Vidal et al. 2013a), and from specialized enumeration techniques (Arnold et al. 2021) to full-blown branch-cut-and-price algorithms (Queiroga et al. 2020).

Run-time and solution quality are intimately connected because a faster approach permits more iterations and therefore gives more chances of improvement. This trade-off is especially visible when observing the evolution of ruin-and-recreate approaches. Early work by Shaw 1998 was oriented towards exact solution approaches (through constraint programming and branch-and-bound) whereas later implementations by Schrimpf et al. 2000; Ropke and Pisinger 2006a; Ropke and Pisinger 2006b; Christiaens and Vanden Berghe 2020 went the opposite way towards fast greedy reconstruction approaches. There is, however, a risk to reconstructing the solutions (i.e., solving the subproblems) with a simple greedy construction algorithm because the probability of producing a good solution decreases quickly with the size of the subproblems. Moreover, with the dramatic improvements of exact algorithms based on branch-cut-and-price for vehicle routing, we are currently witnessing the emergence of new decomposition algorithms exploiting exact methods for the subproblems (Queiroga et al. 2020).

2.4 Utilisation of subproblem solutions

When defining a subproblem supported in an elite solution (Section 2.2), it is possible to directly exploit the result to replace or improve the elite solution. This approach has been adopted in most heuristic decomposition strategies (see, e.g., Groër et al. 2011; Vidal et al. 2013a) because it permits one to quickly benefit from the effort spent on the subproblems. There are, however, other situations in which problem attributes make it more difficult to exploit the results of decomposition. Lahrichi et al. 2015, for example, introduced a sophisticated solution approach for a multi-depot multi-period CVRP, in which solutions of simpler CVRP subproblems are concurrently generated and integrated (i.e., combined together) to form complete solutions of the original problem. The integration step is done with a dedicated optimization algorithm (El Hachemi et al. 2015) which produces a solution of the original problem and aims to retain most of the characteristics of the subproblem solutions.

In conclusion, considering recent work on heuristic decomposition techniques, we recommend to opt for *independent* and *homogeneous* decompositions *supported in elite solutions* because this greatly facilitates the use of the knowledge gained from the decomposition phases. We will follow this approach in the experimental studies of this paper, considering two main classes of decomposition: a partitioning approach that defines subproblems based on the routes of an elite solution (called route-based decomposition), and a coarsening approach that iteratively fixes sequences of customers (called path-based decomposition). For each of these decomposition approaches, we consider different notions of relatedness and different subproblem definitions.

3 Design of Experiments

To perform an experimental analysis of decomposition methods for the CVRP, we used as the underlying solvers two representatives of modern metaheuristics: the ALNS of Pisinger and Ropke 2007 and the HGS of Vidal 2022. By using two frameworks, we guarantee that the observed effects of decomposition methods are not solver-specific. This also permits us to draw more general conclusions and verify that the impact of decomposition is consistently positive across multiple solvers. In the remainder of this section, we describe the decomposition techniques considered in our numerical experiments and the metaheuristics into which they will be integrated. All these decompositions rely on the information of an elite solution, which will be extracted from the metaheuristic.

3.1 Route-based decomposition

Our route-based decomposition methods create a set of independent subproblems $\mathcal{S} = \{S_1, \ldots, S_k\}$, solve each subproblem using a recursive call to the solver, and rebuild a solution to the original problem by merging the solutions to the subproblems. Each subproblem is defined by a pair $S_j = (V_j, p_j)$ where V_j is a subset of customers, and p_j is the number of vehicles assigned to subproblem j. Recall that the geographical coordinates (x_v, y_v) of each location $v \in V$ are known. In addition, we assume that $(x_0, y_0) = (0, 0)$.

We describe a CVRP solution as a set of routes $\mathcal{R} = \{R_1, \ldots, R_p\}$, where each route is given as a sequence of customers $R_i = (0, v_{i1}, \ldots, v_{ir_i}, 0)$. The number of customers visited by vehicle i is r_i. For convenience, we also define the set of customers visited by vehicle i as $W_i = \{v_{i1}, \dots, v_{ir_i}\}$. If a vehicle i is not used, then $W_i = \emptyset$ and $r_i = 0$.

In a route-based decomposition method, the vertex set V_j of each subproblem S_j is built from a subset of the routes in a given solution $\mathcal{R}$. This subset is indexed by $\mathcal{I}_j \subseteq \{1, \dots, p\}$, i.e., $V_j = \bigcup_{i \in \mathcal{I}_j} W_i$. We set the number of vehicles to $p_j = |\mathcal{I}_j|$ to ensure the feasibility of subproblem S_j (provided the given solution $\mathcal{R}$ was feasible). The route-based decomposition methods that we study differ in how the index set $\mathcal{I}_j$ is determined.

We introduce the concept of the *barycenter* of a non-empty route R_i, which is frequently used in the following, and defined as the point in the Euclidean space $\mathbb{R}^2$ with

$$b_i = (b_i^{\mathrm{x}}, b_i^{\mathrm{y}}) = \left(\frac{1}{|V_i|} \sum_{v \in V_i} x_v, \frac{1}{|V_i|} \sum_{v \in V_i} y_v \right).$$

In the following, when using barycenters to create subproblems, we disregard empty routes. The vehicles corresponding to such empty routes, if any, are distributed uniformly among the created subproblems. To increase the chances of finding a high-quality solution to the subproblems, we introduce a parameter $m \in \mathbb{N}$ that denotes a target number of customers to assign to each subproblem.

3.1.1 Random route decomposition

In *random route decomposition*, the routes of $\mathcal{R}$ are shuffled, and the resulting order is used to define subproblems. Starting with $j = 1$, set $\mathcal{I}_j$ is created by adding (the indices of) routes until $\sum_{i \in \mathcal{I}_j} r_i \geq m$ (remember r_i denotes the number of customers in route R_i). Then, subproblem j is "closed", j is incremented by one, and the procedure continues.

3.1.2 Barycenter sweep decomposition

Like *random route decomposition*, this method first sorts the routes and then uses the resulting order to build the subproblems. The difference lies in how the routes are ordered, namely by increasing polar angle $\vartheta_i \in [-\pi, \pi]$ of their barycenter. With $\tau_i = \arctan \left| b_i^{\mathrm{y}} / b_i^{\mathrm{x}} \right|$, then the angle is

$$\vartheta_i = \begin{cases} \tau_i & \text{if } b_i^{\mathrm{x}} \geq 0, b_i^{\mathrm{y}} \geq 0 \\ \pi - \tau_i & \text{if } b_i^{\mathrm{x}} < 0, b_i^{\mathrm{y}} \geq 0 \\ -\tau_i & \text{if } b_i^{\mathrm{x}} \geq 0, b_i^{\mathrm{y}} < 0 \\ -\pi + \tau_i & \text{if } b_i^{\mathrm{x}} < 0, b_i^{\mathrm{y}} < 0. \end{cases}$$

This method was already used in Vidal et al. 2013a.

3.1.3 Quadrant decomposition

This method creates at most four subproblems, one for each quadrant, grouping together routes with the barycenter in the same quadrant. The method does not use parameter m, thus giving less control on the size of the created subproblems. Furthermore, for instances in which the depot lies at the bottom-left corner, the method gives a subproblem which is identical to the original problem. The index sets to perform decomposition are the following (we only use non-empty ones):

$$\mathcal{I}_1 = \{i : b_i^{\mathrm{x}} \geq 0,\ b_i^{\mathrm{y}} \geq 0\}, \qquad \mathcal{I}_2 = \{i : b_i^{\mathrm{x}} < 0,\ b_i^{\mathrm{y}} \geq 0\},$$
$$\mathcal{I}_3 = \{i : b_i^{\mathrm{x}} \geq 0,\ b_i^{\mathrm{y}} < 0\}, \qquad \mathcal{I}_4 = \{i : b_i^{\mathrm{x}} < 0,\ b_i^{\mathrm{y}} < 0\}.$$

The above-described *quadrant decomposition* is related to the one used by Groër et al. 2011, with the difference that the authors divide the Euclidean plane in two halves rather than into four quadrants. They then assign routes to one of the two half-planes if they have at least one customer in the half-plane, leading to overlapping subproblems which need to be solved sequentially. We instead require that the barycenter lies in the quadrant and, therefore, our subproblems do not overlap.

3.1.4 Barycenter clustering decomposition

This method builds $k = \lceil n/m \rceil$ subproblems, by partitioning the routes into k clusters and creating one subproblem for each cluster. To cluster together routes with nearby barycenters, we use the popular k-means algorithm (MacQueen 1967) using the barycenters as points. We use the k-means++ method (Arthur and Vassilvitskii 2007) to generate the initial clustering.

3.1.5 Historical relatedness clustering decomposition

As in the previous method, we want to first build k clusters of routes and then create the subproblems accordingly. What changes here is the clustering criterion: this method groups together routes whose customers have often been visited by the same vehicle in past solutions. When using HGS, we record all solutions produced as offspring after they went through the local search phase. When using ALNS, we use accepted solutions. Let $z_{vu} \in \mathbb{N}$ be the number of times two customers v and u appeared in the same route in a recorded solution. The historical relatedness between two routes $R_i, R_{i'} \in \mathcal{R}$ is defined as

$$Z_{ii'} = \sum_{v \in V_i} \sum_{u \in V_{i'}} z_{vu}.$$

With this metric, routes with higher scores are more related to each other; by contrast, clustering algorithms assume that points within a short distance are more related. Therefore, we use the inverse of $Z_{ii'}$ as the distance between two routes. Unlike the previous method, which used points in the Euclidean space (namely, the barycenters) to represent routes, we now do not have underlying positions characterising the routes. Therefore, we cannot use the k-means algorithm (which relies on such a representation) and instead use the k-medoids algorithm (Park and Jun 2009).

3.1.6 Limitations

The decompositions *barycenter sweep*, *quadrant*, and *barycenter clustering* require that the coordinates of the customers are known; *barycenter clustering* further uses the Euclidean distance within the k-means algorithm. If only a Euclidean distance matrix is available, customer coordinates can be recovered (Young and Householder 1938). Otherwise, if the distance matrix is not Euclidean, *barycenter clustering* can be adapted to use the k-medoid algorithm instead of k-means. In this case, the distance between two routes can be set as the average distance between their visited customers.

3.2 Path-based decompositions

A path-based decomposition method creates a smaller problem by merging groups of customers. We first explain how to merge the customers belonging to a single directed path T. Such a directed path is defined by a sequence of consecutive arcs and denoted by an ordered succession of the customers it visits: $T = (v_1, \ldots, v_{r_T})$. Our goal is to reduce the size of the problem by removing all customers visited by T and replacing them with a single customer, which represents all customers in T visited in the given order $(v_1, \ldots, v_{r_T})$. The new customer v_T has the following characteristics. Its *demand* is the sum of the demands of the customers it replaces, $q_{v_T} = \sum_{v \in T} q_v$. The *travel cost* from (to) a vertex $u \in V \setminus T$ to (from) v_T are, respectively, $c_{uv_T} = c_{uv_1}$ and $c_{v_T u} = c_{v_{r_T} u}$, and its *service time* is $\sum_{i=2}^{r_T} c_{v_{i-1} v_i}$.

The methods perform the decomposition by shrinking all paths of a set $\mathcal{T}$, solving the reduced subproblem, and recovering the corresponding solution for the original problem. Set $\mathcal{T}$ contains paths extracted from a given elite solution. Some of these paths might overlap or be consecutive, in which case we merge them. We also exclude arcs to and from the depot. For example, if the paths extracted from the solution are $(0, v_1, v_2)$ and (v_2, v_3, v_4) then set $\mathcal{T}$ will contain the single path (v_1, v_2, v_3, v_4). A parameter p_{L} determines the length of the paths extracted from the solution. In the previous example, we have $p_{\mathrm{L}} = 2$, i.e., each path contains three vertices. Note that when $p_{\mathrm{L}} = 1$, single arcs are extracted.

Our algorithm solves the reduced subproblem by calling the solver recursively, i.e., it solves the subproblem as a smaller CVRP in which the visiting sequence and direction of shrunken customers is fixed. The subproblem could also be described as a capacitated arc routing problem (see Golden and Wong

1981), fixing the sequence of visits but not the direction, or as a capacitated clustered VRP (see Sevaux and Sörensen 2008; Battarra et al. 2014) not even fixing the sequence of visits. We decided to avoid these options because they would result into an heterogeneous decomposition.

To devise a concrete decomposition method, we must make two decisions: (i) how to select the paths, and (ii) how many paths to select. For the latter decision, we always stop selecting paths as soon as the number of customers in the resulting subproblem reaches our target number of customers m (introduced in Section 3.1). The first decision, how to select paths, differentiates the three methods described in the following.

3.2.1 Path random decomposition

Path random decomposition takes a random sample of all p_{L}-long paths used in the solution. To this end, the algorithm first enumerates all possible paths of length p_{L} and then samples from this set.

3.2.2 Path cost decomposition

This method first sorts all p_{L}-long paths in the solution by their cost (i.e., the sum of the costs of the arcs forming the path). It then starts by adding them to $\mathcal{T}$ from the least to the most expensive one until reaching the stopping criterion described above. The idea behind this method is that short paths have a higher probability of being present in good solutions because they visit customers which are close to each other.

3.2.3 Path relatedness decomposition

This method assigns to each p_{L}-long path T a score $\sum_{i=2}^{r_T} z_{v_{i-1}v_i}$ (where z is defined in Section 3.1.5). The paths are then sorted in descending score order and added to $\mathcal{T}$ until reaching the stopping criterion. With this method, arcs connecting customers which are often consecutive in past solutions are more likely to be selected.

3.3 Integrating decomposition in ALNS

The ALNS metaheuristic framework has proved very useful to solve large instances of routing problems, including the CVRP (Pisinger and Ropke 2019). The algorithm moves from one solution to the next alternatively applying destroy and repair moves. A destroy move removes part of a solution, while a repair move rebuilds a destroyed solution. In our implementation, which is based on that of Pisinger and Ropke (2007), repair moves also perform local search. Because different moves work better with different instances of a problem, the algorithm adaptively learns which moves should be used more often during the current run. To this end, it associates a score to each move. The score increases when the move produces a good solution and decreases otherwise. In each iteration, a destroy and a repair move are selected randomly with a probability proportional to their score (roulette wheel selection). To promote the exploration of different parts of the solution space, the current solution can be replaced by another solution of worse cost. An *acceptance criterion* determines when the newly generated solution should take the place of the incumbent. Following the advice of Santini et al. 2018, we use a Linear Record-to-Record Travel acceptance criterion. Algorithm 1 provides a schematic description of the ALNS framework. For a more detailed description of ALNS for the CVRP, see Santini et al. 2018.

In our implementation, we start the decomposition process once every d iterations (line 7 in Algorithm 1). We choose the current solution as the elite solution to support decomposition. The acceptance criterion that we use guarantees that the current solution is of high quality. Its cost is only allowed to differ from the cost of the best solution by a small margin, which decreases over time.

As we will explain in more detail in Section 4, we use a time limit in the main ALNS process (line 4), but we resort to an iteration limit in the children ALNS processes (line 9). This is because it is hard to estimate in advance how much time a subproblem needs to reach a good solution.

Algorithm 1 Overview of ALNS with decomposition.

Generate an initial solution x^0 (see Pisinger and Ropke 2007, Sec. 5.7)
Set the current solution and the best solution $x \leftarrow x^0$, $x^* \leftarrow x^0$
Set the *iteration* counter $h \leftarrow 0$

while current time $<$ time limit **do**
    Increase the *iteration* counter, $h \leftarrow h + 1$
    Pick a destroy method d and a repair method r, using roulette wheel selection

    if parameter d divides h without remainder **then** ▷ Decomposition phase
        Get subproblems $S_1, \ldots, S_k$ via decomposition based on x
        Solve each subproblem calling ALNS recursively (10000 iterations)
        Build a solution x' from the best solutions of each subproblem
    else
        Get a new solution $x' = r\big(d(x)\big)$

    if *accept* x' **then**
        Update the current solution $x \leftarrow x'$

    if solution x' is the new best **then**
        Update the best solution $x^* \leftarrow x'$

    Update the destroy and repair methods weights

return the best solution x^*

3.4 Integrating decomposition in HGS

HGS is a genetic algorithm (GA) devised for a wide range of VRPs, which mainly differs from classical GAs in its advanced management of population diversity. The algorithm allows feasible and infeasible solutions to coexist in the population (in their respective subpopulations). The fitness of a solution is computed based on its cost, a penalty for possible infeasibilities, and a reward for its contribution to population diversity. We schematically describe HGS in Algorithm 2 and refer the reader to the works of Vidal et al. 2012 and Vidal 2022 for a complete description.

Decomposition appears on line 23 in Algorithm 2 . The algorithm starts a decomposition phase once every d iterations. Because our decomposition methods are supported in an elite solution, we first select randomly an elite individual from the 10% with the lowest cost (line 24). After generating the subproblems with the chosen decomposition method, we call HGS recursively. Analogously to what we do for ALNS, we use a time limit for the main HGS process and an iteration limit for the subproblems.

4 Computational analysis

This section presents a comprehensive numerical study to assess the impact of decomposition methods on the quality of solutions produced by ALNS and HGS. The source code and the instances used are available on-line at `https://github.com/alberto-santini/cvrp-decomposition` (Santini 2022).

We study the following methods:

- No decomposition;
- Route-based methods: random route, barycenter sweep, barycenter clustering, quadrant, historical relatedness clustering.
- Path-based methods: arc random, arc cost, arc history (corresponding to the methods described in Section 3.2 when p_L is 1, i.e., considering arcs); path random, path cost, path history (similar to the previous ones, but p_L is 2). Preliminary experiments showed that further increasing p_L reduced the effectiveness of the method.

Algorithm 2 Overview of HGS with decomposition.

Initialize the feasible and infeasible sub-populations (see Vidal et al. 2012, Sec. 4.6)
Set the *iteration* counter $h \leftarrow 0$
Set the *iterations without improvement* counter to $l \leftarrow 0$

while current time < time limit **do**
  Increase the *iteration* counter, $h \leftarrow h + 1$
  Select two parent individuals P_1, P_2
  Generate a child individual C via crossover of P_1 and P_2
  Execute a local search procedure to improve C

  if C is infeasible **then**
    Place C in the infeasible sub-population
    Try to repair C

  if C is feasible **then**
    Place C in the feasible sub-population
    if C is the new best individual **then**
      Reset *iterations without improvement*, $l \leftarrow 0$
    else
      Increase *iterations without improvement*, $l \leftarrow l + 1$

  if any sub-population reached its maximum size **then**
    Select survivors in this sub-population

  if $l >$ parameter n_{imp} **then**
    Diversify population

  Adapt penalties for infeasibilities

  if parameter d divides h without remainder **then** ▷ Decomposition phase
    Select an elite individual Y
    Get subproblems $S_1, \ldots, S_k$ via decomposition based on Y

    Solve each subproblem calling HGS recursively (1000 generations)
    Build an individual N from the best individual of each subproblem
    Execute a local search procedure to improve N
    Insert N into the appropriate sub-population

return the best feasible individual

For each method, the calibration of two design parameters is required: (i) the target size of the subproblem, $m \in \mathbb{N}$; (ii) the number of iterations between two successive decomposition phases, $d \in \mathbb{N}$;

Initially, we also added the possibility of warmstarting the subproblems. Warmstarting means that we use an initial solution (for ALNS) or an individual in the starting population (for HGS) obtained from the elite solution chosen for decomposition. For example, in a route-based decomposition method, if the algorithm creates a subproblem from routes R_1 and R_2, we can use solution $\{R_1, R_2\}$ directly in the subproblem. During preliminary experiments, we observed that adding such a solution has a considerable adverse impact on the diversity and quality of the subproblem solutions for all decomposition methods. Thus, we deactivated warmstarting.

To calibrate the parameters outlined above on a uniform test bed, we use the largest instances in the "Extended Benchmark" set proposed by Uchoa et al. 2017 (see Section 4.1). These are 20 instances with 600 customers each and with the depot placed at a random point in a containing 100×100 grid. Moreover, 50% of the customers are placed at random, while the other 50% are clustered as described in Uchoa et al. 2017.

In Section 4.2, we compare the performance of the different decomposition methods. As a test set, we use the fifty largest instances of the 100-instance set introduced by Uchoa et al. 2017, which represent a wider diversity of characteristics.

4.1 Sensitivity analysis of the parameters

We run a sensitivity analysis to assess the impact of parameters on the decomposition methods and the relative performance of one method compared to another. We vary the parameters independently using a grid-search approach with $m \in \{50, 100, 150, 200, 250, 300\}$ and $d \in \{1000, 2000, 5000, 7500, 10000\}$. The parameter grid size is 30, and we use ten runs (with different random seeds) for each of the 20 instances and each of the 11 methods.

We use a strict time limit of 1200 seconds as stopping criterion. When given a fixed amount of iterations in the main algorithm instance, using decomposition increases the total running time because of the time spent to solve the subproblems. Furthermore, some decomposition methods require the algorithm to spend more time to solve subproblems than others. For this reason, using a maximum number of iterations would not be a fair way to compare the performance of the different methods.

		Best Configuration						Worst Configuration					
		ALNS			HGS			ALNS			HGS		
Decomposition		m	d	Gap%	m	d	Gap%	m	d	Gap%	m	d	Gap%
No decomposition	N			0.303			0.215			0.303			0.215
Random route	RR	150	1000	0.286	300	7500	0.196	150	5000	0.323	250	1000	0.249
Barycenter sweep	BS	250	1000	0.295	250	5000	0.178	150	5000	0.321	50	1000	0.269
Quadrant	BQ	200	5000	0.297	150	7500	0.202	300	1000	0.316	300	1000	0.263
Barycenter clustering	BC	100	5000	0.277	200	1000	0.166	300	5000	0.303	300	1000	0.205
Historical relatedness	RH	250	10000	0.294	100	7500	0.183	300	1000	0.324	150	1000	0.237
Arc random	RA	300	5000	0.292	150	5000	0.214	200	2000	0.322	300	1000	0.481
Path random	RP	300	2000	0.296	150	7500	0.205	150	1000	0.316	300	1000	0.466
Arc cost	CA	50	10000	0.294	150	10000	0.207	300	10000	0.320	300	1000	0.524
Path cost	CP	100	2000	0.295	100	10000	0.208	100	5000	0.318	300	1000	0.401
Arc history	AH	50	2000	0.289	150	10000	0.198	150	5000	0.318	300	1000	0.506
Path history	PH	300	5000	0.289	50	2000	0.208	150	1000	0.320	300	1000	0.523

Table 1: Parameter configurations yielding the best and worst results for each decomposition method for both ALNS and HGS.

Table 1 reports the results of the analysis. For each decomposition method, under Best Configuration, columns m and d report the best corresponding parameter combination, i.e., the one giving the lowest average gap with respect to the best-known solutions. Column *Gap%* reports the obtained percentage gap. Because no best-known solutions for the *Extended Benchmark* instances are available in the literature, the gap is computed relative to the best solution obtained for each instance across all

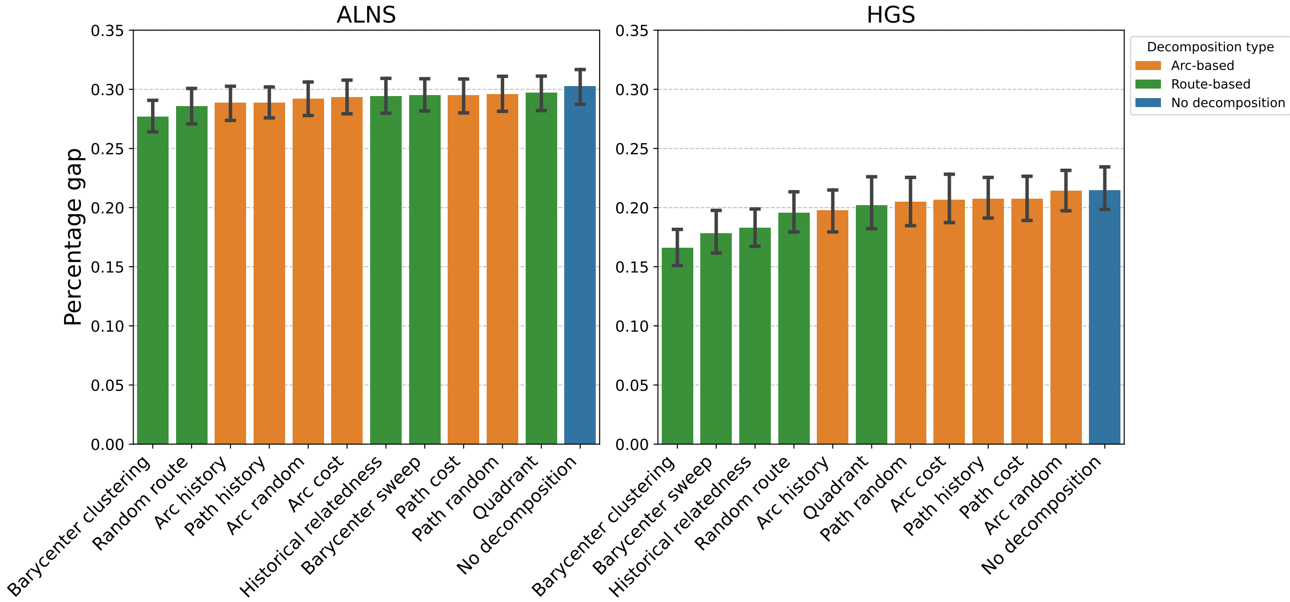


Figure 3: Average percentage gaps to the best-known solutions on the *Extended Benchmark* instances. The results refer to each decomposition method's best parameter combination. Route-based methods are in green, path-based methods in orange, no decomposition is in blue. Error bars denote the standard error of the mean.

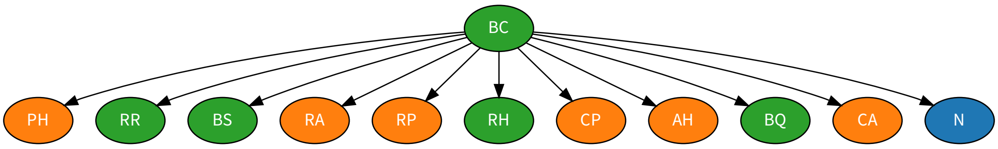
runs of all algorithms. We report under Worst Configuration the worst parameter configurations, i.e., those giving the highest average gaps.

The results show that using decomposition generally improves the quality of the algorithm, independent of the general quality of the base algorithm (HGS or ALNS). With the wrong parameter configuration, however, almost all decompositions give worse results compared to the base algorithm, highlighting the importance of proper parameter tuning. We note that the only decomposition which is competitive even when tuned improperly is *barycenter clustering*.

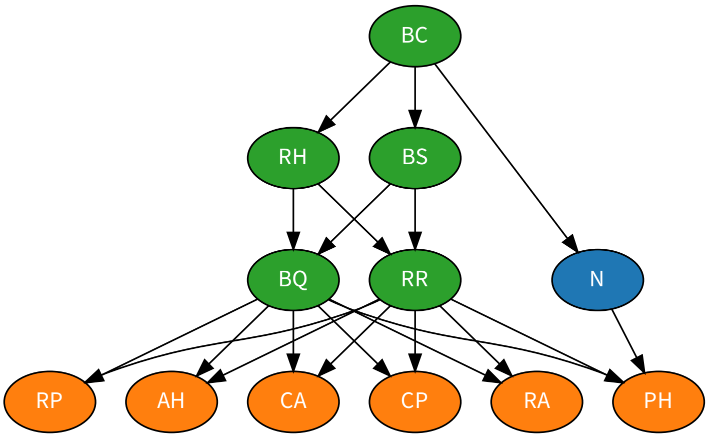
Route-based methods provide the largest improvements, as shown by Figure 3 which reports the average gaps of column *Gap%* of Table 1, with the addition of bars denoting the standard error of the mean. In the figure, route-based methods are in green, path-based ones in orange, and no decomposition is in blue. The figure also shows that HGS benefits proportionally more from the best-performing decomposition methods compared to ALNS.

Because the error bars are overlapping for many of the methods, we confirm the validity of the obtained ranking by running Tukey's Honestly Significant Difference (HSD) test (Tukey 1949) between all pairs of decomposition methods. The data associated to each method is a vector of the same size as the number of instances used, in which each entry gives the average Gap% for one instance across all reruns. Figure 4 reports the results of these tests where each method is represented by an oval with the same color-coding described before. An arrow between two methods indicates that the method at the tail has lower gaps than the method at the head, and that the difference is significant (p-value smaller than 0.05, where the null hypothesis is that the two average gaps are the same). The figure shows that *barycenter clustering* dominates all other decomposition methods when used within ALNS and is the only dominating method. When applied to HGS, *barycenter clustering* still dominates all other methods. Moreover, *historical relatedness* and *barycenter sweep* dominate the other methods, with the exception of *batycenter clustering* and *no decomposition*. For both ALNS and HGS, many methods do not dominate each other, nor do they dominate *no decomposition*.

As these results refer to the same instances on which we tuned the parameters, we cannot rule out the effects of overfitting. For this reason, we defer a detailed analysis of the merits of the single decomposition methods to Section 4.2, in which we test the methods on a different instance set.



(a) Results for ALNS



(b) Results for HGS

Figure 4: Results of Tukey's HSD tests between each pair of methods on the *Extended benchmark* instances. An arrow between two methods indicates that the method at the tail has lower gaps than the method at the head, and that the difference is significant (p-value smaller than 0.05). Route-based methods are in green, path-based methods in orange, no decomposition is in blue.

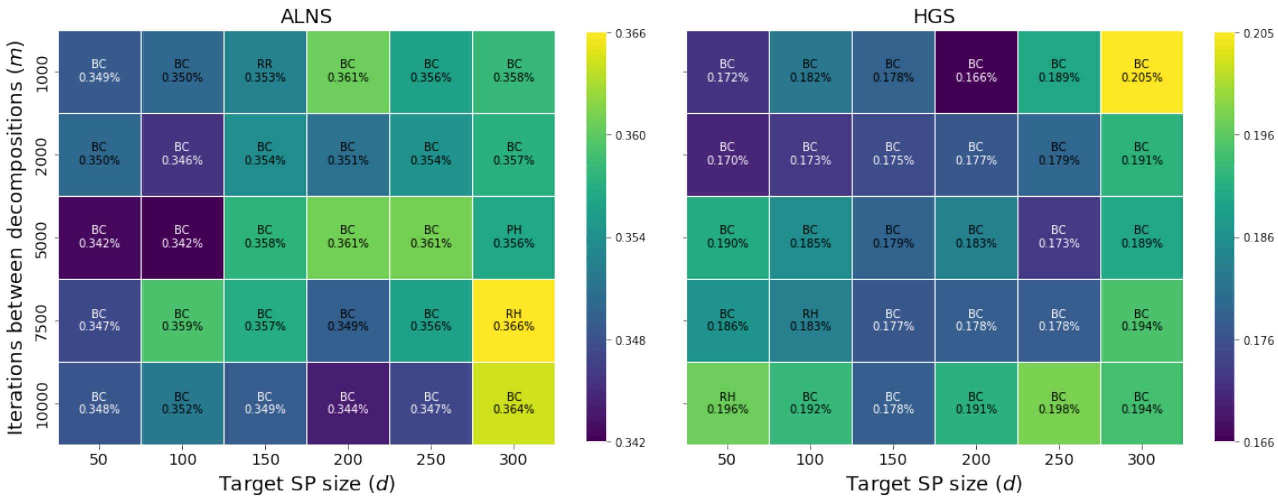


Figure 5: Best algorithm and corresponding percentage gap, for each combination of parameters m (on the y axis) and d (on the x axis). The size of the gaps is given in the color legend.

In the rest of this section, we examine how robust the methods are with respect to the variation of parameters m and d. Figure 5 shows how the best decomposition method changes when moving in the parameter space defined by m (on the y axis) and d (on the x axis). The size of the gaps is given in the color legend; the abbreviations used are the ones listed in Table 1. *Barycenter clustering* performs best for most parameter combinations. Good combinations (darker colors) cluster around the best one and all correspond to the *barycenter clustering* decomposition. Therefore, a user applying decomposition to solve a large-scale routing problem can be sufficiently confident that *barycenter clustering* will improve solution quality, and the user can rely on the method being not too sensitive to parameters m and d.

4.2 Performance comparison of the decomposition methods

To compare the performance of the different methods, we fix the parameters of each method to the best configuration values reported in Table 1, and we conduct experiments on the fifty largest instances of Uchoa et al. 2017, which include between 335 and 1000 customers. Because this instance set is disjoint from the one used in Section 4.1, we mitigate the effects of overfitting parameter values to specific instances. The stopping criterion remains set to a fixed time limit of 1200 seconds. We compute percentage gap with respect to the best-known solutions published by Xavier et al. 2021 as of the 22$^{\text{nd}}$ of December, 2021.

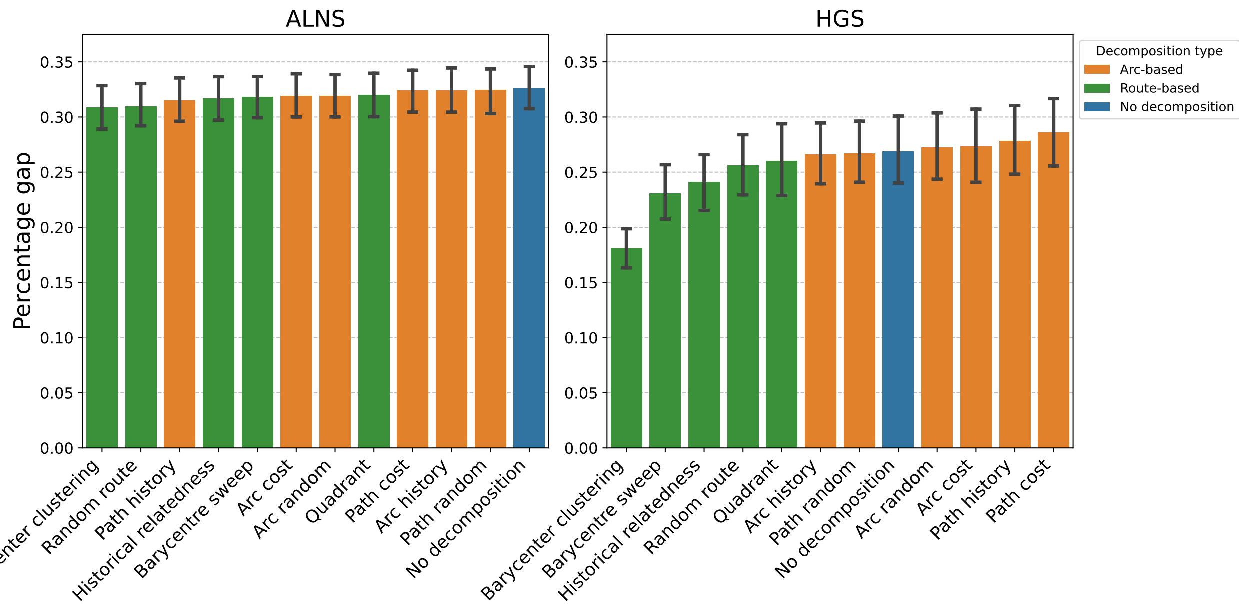
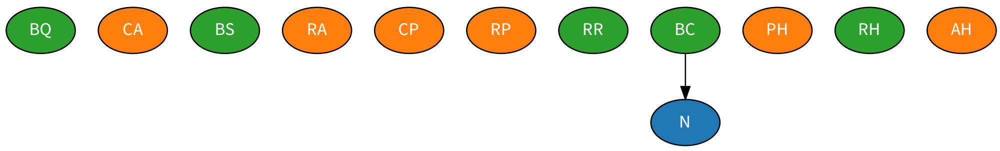


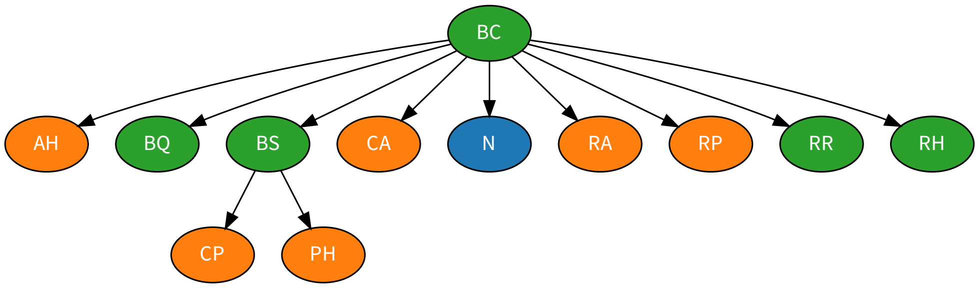
Figure 6: Average percentage gaps to the best-known solution on Uchoa et al. 2017's fifty largest instances. We use each decomposition method with the parameters given in Section 4.1. Route-based methods are in green, path-based methods in orange, no decomposition is in blue. Error bars denote the standard error of the mean.

Figure 6 shows the results on this second set of instances, and the result of the corresponding Tukey's HSD test are reported in Figure 7. Percentage gaps are higher on this test set than they were on the instances used for parameter tuning. The results show that, again, route-based methods outperform path-based methods. In fact, for the HGS algorithm, four path-based methods give higher average gaps compared to *no decomposition*. *Barycenter clustering* shows the best performance of all methods, when applied to both ALNS and HGS. For HGS, the standard error bar of the average for *barycenter clustering* does not even overlap with the bar of the second-best decomposition method, *barycenter sweep*. The HSD analysis further shows that *barycenter clustering* dominates all other methods when applied to HGS. The other methods do not dominate each other, except for *barycenter sweep*, which is superior to both *path history* and *path cost*. Statistical dominance is less clear for ALNS, as in this case *barycenter clustering* only dominates *no decomposition*, while the other methods do not dominate each other.

It should be emphasized that these findings are not mere updates or slight adjustments of knowledge from the literature. In fact, route-based and path-based methods have been used in the literature to similar extent. To our knowledge, the best-performing method, *barycenter clustering*, has not been previously used in the literature. We also highlight that, although the impact of decomposition methods



(a) Results for ALNS.



(b) Results for HGS.

Figure 7: Results of Tukey's HSD tests between each pair of methods on Uchoa et al. 2017's fifty largest instances. An arrow between two methods indicates that the method at the tail has lower gaps than the method at the head, and that the difference is significant (p-value smaller than 0.05). Route-based methods are in green, path-based methods in orange, no decomposition is in blue.

on the gaps to the best-known solutions may appear small, it is relevant for state-of-the-art algorithms, which usually compete within tenths of a percentage point on standard benchmark sets.

Another interesting research question is how the impact of decomposition evolves during the course of the search: how much does decomposition help, and is its contribution the same at the beginning and at the end of the search? To answer this question, we focus on algorithm HGS and the best decomposition method, *barycenter clustering*, and compare it with *no decomposition*. Each time the best individual in the population changes (i.e., the algorithm finds a new best solution), we record the current time and if the individual comes from a standard GA operation (crossover, mutation, local search), or from the decomposition method. This allows us to measure precisely the contribution of the decomposition method in finding new best solutions, and how this contribution varies at different stages of the solution process.

Figure 8 shows our findings. The x axis reports the run-time, divided in ten intervals from 0–10% (corresponding to the first 120 seconds) to 90–100%. The y axis, in logarithmic scale, reports the gap decrease achieved in each intervals. Solid blue bars refer to the pure GA algorithm with no decomposition. Orange bars refer to the algorithm using *barycenter clustering* and are made up of two parts. The bottom part reports the gap decrease from GA operations, while the top part (marked with diagonal hatches) corresponds to the decrease due to decomposition. Note that, in most intervals, the decrease due to pure GA operations is higher for *no decomposition* because more time is spent in GA iterations. The algorithm using *barycenter clustering*, however, generally achieves greater gap reductions thanks to the decomposition phase. Furthermore, the contribution of the decomposition method is proportionally larger in later intervals than at the beginning: when standard GA operations start to achieve increasingly marginal returns, solving decomposed subproblems is still a good time investment.

This last observation, however, prompts another question: would decomposition still be effective if we only applied it as a post-processing step at the end of HGS? To answer this question, we applied

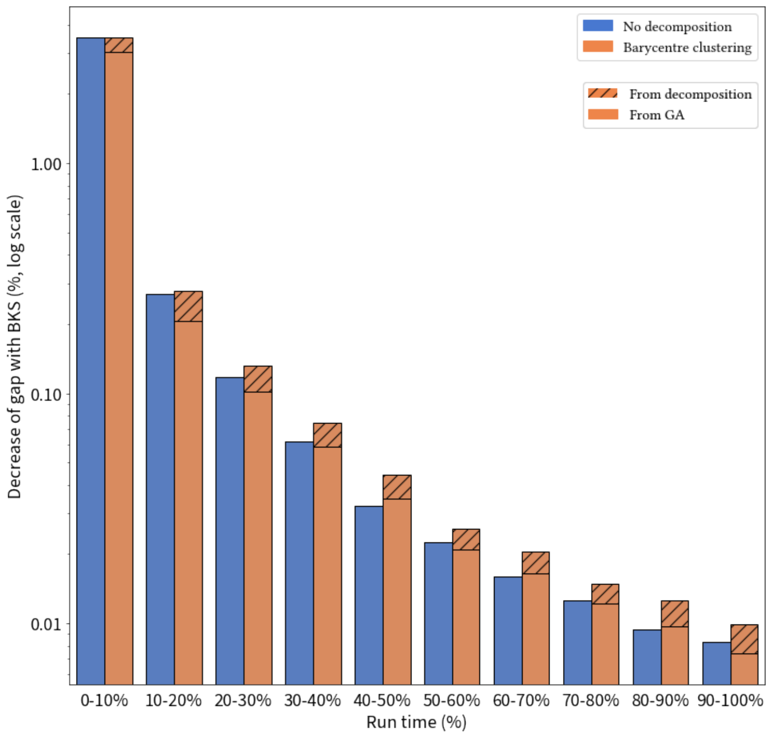


Figure 8: Solution improvement progress for *barycenter clustering* and *no decomposition* used in the HGS algorithm. The CPU time has been divided into ten time intervals of 120 s, reported on the x axis. The values on the y axis (in log scale) denote the average gap improvement over the time intervals. The bars for barycenter decomposition have two parts: the one in solid color marks the contribution to gap decrease coming from standard GA operations (i.e., from cross-over, mutation, local search), the one with diagonal hatches marks the contribution coming from solving the decomposed subproblem (i.e., when the subproblem solution was better than the previous best).

barycenter clustering to the best solution found by HGS with *no decomposition*. The results of this method, which we denote *barycenter clustering**, are reported in Figure 9. Applying the decomposition only at the end of HGS is statistically indistinguishable from *no decomposition*. Therefore, an effective decomposition must be used throughout the execution of the main heuristic algorithm.

5 Conclusions

In this paper, we have reviewed the main characteristics of heuristic decomposition methods for solving large-scale VRPs and related them to key studies. We implemented a large variety of decomposition methods and conducted a systematic comparison of their performance on the CVRP benchmark set of Uchoa et al. 2017 using two state-of-the-art algorithms. According to our experimental results, route-based decomposition methods generally appeared as superior to path-based methods, and *barycenter clustering* (newly proposed in this paper) achieved the overall best performance and led to significant gains.

This analysis has permitted us to perform a structured review and comparison of many classical and new decomposition techniques, and it will be especially valuable to guide researchers and practitioners working on the solution of large-scale VRP instances. Nevertheless, research on heuristic decompositions remains at an early stage, and many additional developments are possible. In particular, the use of more sophisticated learning mechanisms (Arnold and Sörensen 2018) and sparsification techniques (Joshi et al. 2019; Taillard and Helsgaun 2019) could permit a better definition of subproblems and lead to major improvements. Machine learning techniques could also be instrumental to quickly solve many subproblems of moderate size (Kool et al. 2021). Heterogeneous path-based decompositions (e.g., using free visit orientation for the aggregated sequences of visits) can lead to additional improvements on

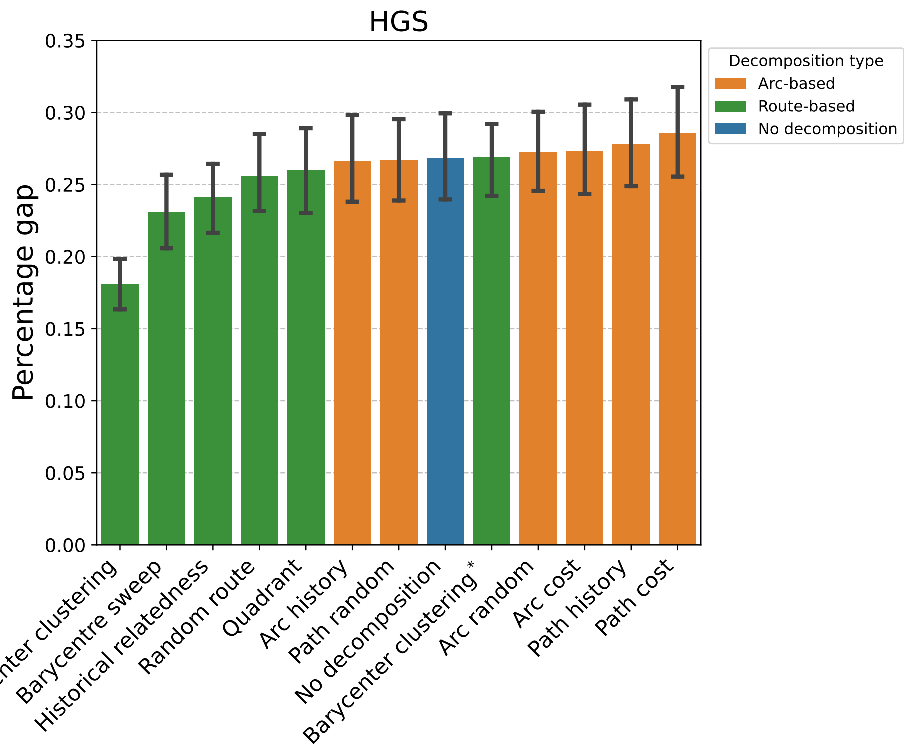


Figure 9: Average percentage gaps to the best-known solution, on Uchoa et al. 2017's fifty largest instances and using the HGS algorithm. This figure adds method Barycenter clustering* to the decompositions presented in Figure 6.

the CVRP. Last but not least, the significant progress of mathematical programming algorithms for the CVRP and its subproblems suggests re-opening research lines connected to the design of efficient matheuristics built on problem decompositions (e.g., as in Queiroga et al. 2020).

Acknowledgments

The authors are grateful to Stefan Ropke for sharing the source code of the ALNS heuristic.

Alberto Santini was partially funded by MICINN (Spain) through the program *Juan de la Cierva Formación*, the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement 945380, ESSEC Business School (France) through the *Visiting Professor* program. The research of Daniele Vigo has been supported by Ministero dell'Istruzione, dell'Università e della Ricerca, Italy under grant PRIN 2015JJLC3E_002, and by USAF under grant FA9550-17-1-0234. The research of Thibaut Vidal has been supported in part by CAPES, by CNPq under grant 308528/2018-2, and by FAPERJ under grant E-26/202.790/2019 in Brazil. This support is gratefully acknowledged.

References

Arnold, Florian, Michel Gendreau, and Kenneth Sörensen (2019). "Efficiently solving very large-scale routing problems". In: *Computers & Operations Research* 107, pp. 32–42. DOI: `10.1016/j.cor.2019.03.006`.

Arnold, Florian, Ítalo Santana, Kenneth Sörensen, and Thibaut Vidal (2021). "PILS: Exploring high-order neighborhoods by pattern mining and injection". In: *Pattern Recognition* 216. DOI: `10.1016/j.patcog.2021.107957`.

Arnold, Florian and Kenneth Sörensen (2018). "What makes a VRP solution good? The generation of problem-specific knowledge for heuristics". In: *Computers & Operations Research* 106, pp. 280–288. DOI: `10.1016/j.cor.2018.02.007`.

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: The advantages of careful seeding". In: *Proceedings of the Eighteenth annual ACM-SIAM symposium on Discrete Algorithms* (New Orleans, Lousiana, USA, Jan. 7–9, 2007). Ed. by Harold Gabow. New Orleans, USA: SIAM, pp. 1027–1035.

Battarra, Maria, Günes Erdoğan, and Daniele Vigo (2014). "Exact algorithms for the clustered vehicle routing problem". In: *Operations Research* 62.1, pp. 58–71. DOI: 10.1287/opre.2013.1227.

Bent, Russell and Pascal Van Hentenryck (2010). "Spatial, temporal, and hybrid decompositions for large-scale vehicle routing with time windows". In: *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. Ed. by David Cohen. Springer, pp. 99–113. DOI: 10.1007/978-3-642-15396-9_11.

Bulhões, Teobaldo, Minh Hoang Ha, Rafael Martinelli, and Thibaut Vidal (2018). "The vehicle routing problem with service level constraints". In: *European Journal of Operational Research* 265.2, pp. 544–558. DOI: 10.1016/j.ejor.2017.08.027.

Chevalier, Cédric and Ilya Safro (2009). "Comparison of coarsening schemes for multilevel graph partitioning". In: *Proceedings of the International Conference on Learning and Intelligent Optimization* (Trento, Italy, Jan. 14–18, 2019). Ed. by Thomas Stützle. Springer. Trento, Italy, pp. 191–205. DOI: 10.1007/978-3-642-11169-3_14.

Christiaens, Jan and Greet Vanden Berghe (2020). "Slack Induction by String Removals for Vehicle Routing Problems". In: *Transportation Science* 54.2, pp. 417–433. DOI: 10.1287/trsc.2019.0914.

Costa, Luciano, Claudio Contardo, and Guy Desaulniers (2019). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". In: *Transportation Science* 53.4, pp. 946–985. DOI: 10.1287/trsc.2018.0878.

El Hachemi, Nizar, Teodor Gabriel Crainic, Nadia Lahrichi, Walter Rei, and Thibaut Vidal (2015). "Solution integration in combinatorial optimization with applications to cooperative search and rich vehicle routing". In: *Journal of Heuristics* 21.5, pp. 663–685. DOI: 10.1007/s10732-015-9296-z.

Geoffrion, Arthur (1970a). "Elements of large-scale mathematical programming. Part I: Concepts". In: *Management Science* 16.11, pp. 652–675. DOI: 10.1287/mnsc.16.11.652.

Geoffrion, Arthur (1970b). "Elements of large-scale mathematical programming. Part II: Synthesis of algorithms and bibliography". In: *Management Science* 16.11, pp. 676–691. DOI: 10.1287/mnsc.16.11.676.

Goeke, Dominik, Timo Gschwind, and Michael Schneider (2019). "Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier". In: *Discrete Applied Mathematics* 264, pp. 43–61. DOI: 10.1016/j.dam.2018.10.015.

Golden, Bruce and Richard Wong (1981). "Capacitated arc routing problems". In: *Networks* 11.3, pp. 305–315. DOI: 10.1002/net.3230110308.

Groër, Chris, Bruce Golden, and Edward Wasil (2009). "The consistent vehicle routing problem". In: *Manufacturing & Service Operations Management* 11.4, pp. 630–643. DOI: 10.1287/msom.1080.0243.

Groër, Chris, Bruce Golden, and Edward Wasil (2011). "A parallel algorithm for the vehicle routing problem". In: *INFORMS Journal on Computing* 23.2, pp. 315–330. DOI: 10.1287/ijoc.1100.0402.

Joshi, Chaitanya, Thomas Laurent, and Xavier Bresson (2019). "An efficient graph convolutional network technique for the travelling salesman problem". In: *Working paper, arXiv*. arXiv: 1906.01227. URL: https://arxiv.org/abs/1906.01227.

Kool, Wouter, Herke van Hoof, Joaquim Gromicho, and Max Welling (2021). "Deep Policy Dynamic Programming for Vehicle Routing Problems". In: *Working paper, arXiv*. arXiv: 2102.11756. URL: http://arxiv.org/abs/2102.11756.

Lahrichi, Nadia, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei, Gloria Cerasela Crişan, and Thibaut Vidal (2015). "An integrative cooperative search framework for multi-decision-attribute combinatorial optimization: Application to the MDPVRP". In: *European Journal of Operational Research* 246.2, pp. 400–412. DOI: 10.1016/j.ejor.2015.05.007.

MacQueen, James (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Ed. by Lucien Lecam and Jerzy Neyman. Vol. 1. Oakland, CA, USA: University of California Press, pp. 281–297.

Park, Hae-Sang and Chi-Hyuck Jun (2009). “A simple and fast algorithm for K-medoids clustering”. In: *Expert Systems with Applications* 36.2, pp. 3336–3341. DOI: 10.1016/j.eswa.2008.01.039.

Pecin, Diego, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa (2017). “Improved branch-cut-and-price for capacitated vehicle routing”. In: *Mathematical Programming Computation* 9, pp. 61–100. DOI: 10.1007/s12532-016-0108-8.

Pessoa, Artur, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck (2019). “A Generic Exact Solver for Vehicle Routing and Related Problems”. In: *Integer Programming and Combinatorial Optimization*. Ed. by Andrea Lodi and Viswanath Nagarajan. Springer, pp. 354–369. DOI: 10.1007/978-3-030-17953-3_27.

Pisinger, David and Stefan Ropke (2007). “A general heuristic for vehicle routing problems”. In: *Computers & Operations Research* 34.8, pp. 2403–2435. DOI: j.cor.2005.09.012.

Pisinger, David and Stefan Ropke (2019). “Large Neighborhood Search”. In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. 3rd. Springer. Chap. 4, pp. 99–127. DOI: 10.1007/978-3-319-91086-4.

Queiroga, Eduardo, Ruslan Sadykov, and Eduardo Uchoa (2020). *A modern POPMUSIC matheuristic for the capacitated vehicle routing problem*. Tech. rep. Caderno do LOGIS, Universidade Federal Fluminense.

Ribeiro, Marcos Henrique, Alexandre Plastino, and Simone Martins (2006). “Hybridization of GRASP metaheuristic with data mining techniques”. In: *Journal of Mathematical Modelling and Algorithms* 5.1, pp. 23–41. DOI: 10.1007/s10852-005-9030-1.

Rodrigues de Holanda Maia, Marcelo, Alexandre Plastino, and Puca Huachi Vaz Penna (2020). “MineReduce: An approach based on data mining for problem size reduction”. In: *Computers and Operations Research* 122, p. 104995. DOI: 10.1016/j.cor.2020.104995.

Ropke, Stefan and David Pisinger (2006a). “A unified heuristic for a large class of vehicle routing problems with backhauls”. In: *European Journal of Operational Research* 171.3, pp. 750–775. DOI: 10.1016/j.ejor.2004.09.004.

Ropke, Stefan and David Pisinger (2006b). “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”. In: *Transportation science* 40.4, pp. 455–472. DOI: 10.1287/trsc.1050.0135.

Rossit, Diego Gabriel, Daniele Vigo, Fernando Tohmé, and Mariano Frutos (2019). “Visual attractiveness in routing problems: a review”. In: *Computers & Operations Research* 103, pp. 13–34. DOI: 10.1016/j.cor.2018.10.012.

Santini, Alberto (2019). “An adaptive large neighbourhood search algorithm for the orienteering problem”. In: *Expert Systems with Applications* 123, pp. 154–167. DOI: 10.1016/j.eswa.2018.12.050.

Santini, Alberto (2022). “Repository cvrp-decomposition”. In: *GitHub*. DOI: 10.5281/zenodo.6097671. URL: https://github.com/alberto-santini/cvrp-decomposition.

Santini, Alberto, Stefan Ropke, and Lars Magnus Hvattum (2018). “A comparison of acceptance criteria for the Adaptive Large Neighbourhood Search metaheuristic”. In: *Journal of Heuristics* 24 (5), pp. 783–815. DOI: 10.1007/s10732-018-9377-x.

Schrimpf, Gerhard, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck (2000). “Record breaking optimization results using the ruin and recreate principle”. In: *Journal of Computational Physics* 159.2, pp. 139–171. DOI: 10.1006/jcph.1999.6413.

Sevaux, Marc and Kenneth Sörensen (2008). “Hamiltonian paths in large clustered routing problems”. In: *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing*. Ed. by Caroline Prodhon, Roberto Wolfler-Calvo, Labadi Nacima, and Christian Prins, pp. 411–417.

Shaw, Paul (1997). “A new local search algorithm providing high quality solutions to vehicle routing problems”. In: *Technical report, Department of Computer Science, University of Strathclyde*.

Shaw, Paul (1998). “Using constraint programming and local search methods to solve vehicle routing problems”. In: *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. Ed. by Michael Maher and Jean-Francois Puget. Springer, pp. 417–431. DOI: 10.1007/3-540-49481-2_30.

Taillard, Eric (1993). “Parallel iterative search methods for vehicle routing problems”. In: *Networks* 23.8, pp. 661–673. DOI: 10.1002/net.3230230804.

Taillard, Eric and Keld Helsgaun (2019). "POPMUSIC for the travelling salesman problem". In: *European Journal of Operational Research* 272.2, pp. 420–429. DOI: 10.1016/j.ejor.2018.06.039.

Toth, Paolo and Daniele Vigo, eds. (2014). *Vehicle Routing: Problems, Methods, and Applications*. MOS-SIAM Series on Optimization. SIAM.

Tukey, John (1949). "Comparing individual means in the analysis of variance". In: *Biometrics* 5.2, pp. 99–114. DOI: 10.2307/3001913.

Uchoa, Eduardo, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian (2017). "New benchmark instances for the capacitated vehicle routing problem". In: *European Journal of Operational Research* 257.3, pp. 845–858. DOI: 10.1016/j.ejor.2016.08.012.

Vidal, Thibaut (2017). "Node, Edge, Arc Routing and Turn Penalties: multiple Problems–One Neighborhood Extension". In: *Operations Research* 65.4, pp. 992–1010. DOI: 10.1287/opre.2017.1595.

Vidal, Thibaut (2022). "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood". In: *Computers & Operations Research* 140. DOI: 10.1016/j.cor.2021.105643.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei (2012). "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems". In: *Operations Research* 60.3, pp. 611–624. DOI: 10.1287/opre.1120.1048.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins (2013a). "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows". In: *Computers & operations research* 40.1, pp. 475–489. DOI: 10.1016/j.cor.2012.07.018.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins (2013b). "Heuristics for multi-attribute vehicle routing problems: a survey and synthesis". In: *European Journal of Operational Research* 231.1, pp. 1–21. DOI: 10.1016/j.ejor.2013.02.053.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins (2014). "A unified solution framework for multi-attribute vehicle routing problems". In: *European Journal of Operational Research* 234.3, pp. 658–673. DOI: 10.1016/j.ejor.2013.09.045.

Vidal, Thibaut, Gilbert Laporte, and Piotr Matl (2020). "A concise guide to existing and emerging vehicle routing problem variants". In: *European Journal of Operational Research* 286 (2), pp. 401–416. DOI: 10.1016/j.ejor.2019.10.010.

Walshaw, Chris (2002). "A multilevel approach to the travelling salesman problem". In: *Operations research* 50.5, pp. 862–877. DOI: 10.1287/opre.50.5.862.373.

Xavier, Ivan, Daniel Oliveira, and Eduardo Queiroga (2021). *CVRPLIB – All instances, Accessed December 12, 2021.* Algorithms, Optimisation and Simulation Group, PUC Rio. URL: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/ (visited on 12/22/2021).

Young, Gale and Alston Scott Householder (1938). "Discussion of a set of points in terms of their mutual distances". In: *Psychometrika* 3, pp. 19–22. DOI: 0.1007/BF02287916.