A lexicographic minimax approach to the vehicle routing problem with route balancing

Fabien Lehuédé, Olivier Péton, Fabien Tricoire

A lexicographic minimax approach to the vehicle routing problem with route balancing

Fabien Lehuédé[1], Olivier Péton[1], Fabien Tricoire[2,3]
1. IMT Atlantique, Laboratoire des Sciences du Numérique de Nantes (LS2N, UMR CNRS 6004), Nantes, France
2. Institute for Production and Logistics Management, Johannes Kepler University Linz, Austria
3. Department of Business Decisions and Analytics, University of Vienna, Austria

Abstract

Vehicle routing problems generally aim at designing routes that minimize transportation costs. However, in practical settings, many companies also pay attention at how the workload is distributed among its drivers. Accordingly, two main approaches for balancing the workload have been proposed in the literature. They are based on minimizing the duration of the longest route, or the difference between the longest and the shortest routes, respectively. Recently, it has been shown on several occasions that both approaches have some flaws. In order to model equity we investigate the lexicographic minimax approach, which is rooted in social choice theory. We define the leximax vehicle routing problem which considers the bi-objective optimization of transportation costs and of workload balancing. This problem is solved by a heuristic based on the multi-directional local search framework. It involves dedicated large neighborhood search operators. Several LNS operators are proposed and compared in experimentations.

Keywords: vehicle routing problem, workload balancing, equity in route duration, multi-directional local search, large neighborhood search

1. Introduction

Most vehicle routing problems (VRPs) studied in the operations research literature deal with the design of a set of routes of minimal cost to serve a set of customers. In many practical cases, companies seek cost minimization as well as the optimization of criteria related to vehicles and drivers. In particular, preserving equity among drivers through a good balance of their workload is often sought. The workload of a driver (or a vehicle) can be measured by the length, cost or duration of the route assigned to this driver. The problem of designing a set of balanced routes has been generally studied in a multi-objective setting and known under the name VRP with *route balancing* (Borgulya, 2008; Jozefowiez et al., 2009; Lacomme et al., 2015) *load balancing* (Lee and Ueng, 1999) or *route balance* (Mandal et al., 2015; Halvorsen-Weare and Savelsbergh, 2016).

Historically, two main *equity measures* have been proposed in the VRP literature. First, *min-max approaches*, as in Golden et al. (1997), aim at minimizing the load of the most loaded driver. An

☆Paper accepted for publication in EJOR, https://doi.org/10.1016/j.ejor.2019.09.010.

unfortunate consequence of this approach is that all solutions with the same value for the longest route have the same equity measure. Therefore, significantly different solutions are considered equivalent. The min-max approach is further investigated in Bertazzi et al. (2015), which analyzes min-max optimal solutions with respect to minimizing cost.

The second equity measure considers the difference between the longest route and the shortest route. This approach was first introduced by Lee and Ueng (1999) with a motivation to integrate employee's welfare in optimization models. This work was followed by several other contributions in the past twenty years (see e.g. Jozefowiez et al. (2002, 2009); Mandal et al. (2015); Lacomme et al. (2015)). A drawback of this measure is its non-monotonicity: it can be improved by increasing, even in an inconsistent way, the load of the shortest route of a solution without decreasing the load of the others.

Example 1. *Consider a VRP with three routes, and three solutions* s_1, s_2 *and* s_3 *with route durations vectors* $(8, 7, 7)$*,* $(8, 6, 7)$ *and* $(10, 10, 10)$ *respectively.*

According to the min-max approach, solutions s_1 *and* s_2 *are equivalent, whereas* s_2 *should be considered better because one driver is less loaded than in* s_1 *and the two other routes require the same amount of work.*

According to the second equity measure, solution s_1 *with route durations* $(8, 7, 7)$ *offer a better workload balance than* s_2 *with route durations* $(8, 6, 7)$*. This is mathematically true, but not really consistent with employee's welfare considerations. Note also that solution* s_3*, with route durations* $(10, 10, 10)$*, is perfectly balanced, and thus would have a better equity measure than* s_1 *and* s_2*.*

The recent survey and analysis of Matl et al. (2017) provides a deep analysis of equity measures for the VRP in a bi-objective context, where the first objective is the solution cost. In Matl et al. (2017), the difference between the longest route and the shortest route is called *range*. The article includes an axiomatic approach that involves various equity measures and lists the good properties that should be verified by an equity measure. Hence, among others, *range* is compared to the min-max and lexicographic minimax ordering approaches. The authors introduce the notion of *inconsistency*, which characterizes solutions like s_1 and s_3 that could be improved in terms of load and cost, at the price of deteriorating the workload balancing measure.

It can be argued that in a bi-objective (cost + equity measure) setting, the set of Pareto-optimal solutions should only contain solutions that make sense in practice. This leads to the notion of *TSP-optimality*: a route is called TSP-optimal if re-ordering its customers cannot result in a decrease of its duration. Among many important observations, Matl et al. (2017) observe that, even if TSP-optimality is enforced for each tour: (i) the Pareto set contains only 30% of consistent solutions, and (ii) inconsistent solutions can dominate some consistent solutions. This is observed for all equity measures that are not monotonic with respect to route duration. In addition, the authors show that the lexicographic minimax approach finds all consistent solutions and no inconsistent solution.

Using the range objective can also lead to poor assignment of customers to routes, in order to produce solutions that are balanced by that measure. In a recent computational study where several route balance models are compared, Halvorsen-Weare and Savelsbergh (2016) provide examples of so-called *artificially balanced* solutions that are considered to be good when using the range objective. To tackle this problem, procedures have been proposed (see e.g. Jozefowiez et al. (2002)) to avoid creating routes that are not *TSP-optimal*, but bad assignment of customers to routes remains a problem.

In this paper, we use the lexicographic minimax approach as an equity measure. The lexicographic minimax approach is used to model equity in many other fields (Ogryczak et al., 2014). To our knowledge, it has been integrated in VRP algorithms in only one paper: Saliba (2006) presents adaptations of sequential insertion and savings heuristics in parameterized versions, as well as adaptations of the 2-Opt, string exchange and string relocation local search operators for a CVRP where the lexicographic minimax approach is applied in a single-objective fashion. These heuristics are compared on instances from the VRP-Lib. For an exhaustive recent review of all contributions in VRP on workload balancing, we refer to the recent papers of Halvorsen-Weare and Savelsbergh (2016) and Matl et al. (2017).

A driver working time being directly related to its vehicle route duration, we consider that the workload of a route is its duration. This is also the most common metric used to evaluate a route. From Matl et al. (2017), *"most works consider tour length as the equity metric, based either on distance or duration"*. Among the 35 vehicle routing publications incorporating equity concerns in this survey, all but 3 take a length measure of the route as a metric for its workload. However, our approach is not tied to this definition and can be applied to other route load metrics.

Section 2 introduces the *leximax*-VRP, which integrates workload balancing following the lexicographic minimax approach. To tackle this problem, we propose a multi-directional local search approach in Section 3, including the Large Neighborhood Search (LNS) operators specifically designed to guide the search towards balanced solutions. Finally, we present several experiments to evaluate the proposed approaches as well as an extensive analysis of the solutions that result from this optimization.

2. The lexicographic minimax approach to the capacitated vehicle routing problem

The lexicographic minimax refines the min-max approach (Dubois et al., 1996): informally speaking, when a minimal value has been found for the longest route, the lexicographic minimax considers the second longest route, the third longest route, and so on, until all ties have been broken. Hence, in this approach, solution s_1 with route durations $(8, 6, 7)$ strictly dominates s_2 with route durations $(8, 7, 7)$, which in turns dominates s_3 with route durations $(10, 10, 10)$.

As summarized by Ogryczak et al. (2014) in a survey on *fair optimization and networks*, the lexicographic minimax approach has been used in operations research in applications such as bandwidth allocation and network optimization (see e.g. Ogryczak et al. (2005); Nace and Orlin (2007); Radunović and Boudec (2007); Luss (2010)), facility location (Ogryczak, 1997), supply chain optimization (Liu and Papageorgiou, 2013), air traffic flow management (Bertsimas et al., 2008) to produce *equitable* (or indifferently called *fair*) solutions. This approach is related the *leximax* ordering (or rather *leximin* in a welfare maximization context) in social choice theory, introduced by Sen (1970). It has also been widely used under this name in decision making (Dubois et al., 1996) or constraint programming (see e.g. Dubois et al. (1995); Bouveret and Lemaître (2009)). Other definitions of the same concept includes min-max fairness (see e.g. Radunović and Boudec (2007); Ogryczak et al. (2014)), lexicographic min-max optimization (Ogryczak and Śliwiński, 2006), lexicographic max-ordering (Ehrgott, 1998; Saliba, 2006), lexmaxmin (Behringer, 1981).

We refer to Karsu and Morton (2015) for a review of fair optimization approaches and Moulin (2004) for a discussion on fairness in various contexts. Bertsimas et al. (2011) present theoretical results on the price of fairness.

In our lexicographic minimax approach to the vehicle routing problem with route balancing, we use the following definition of the leximax ordering, adapted from the leximin definition of

Bouveret and Lemaître (2009) and Sen (1970).

Definition 1. *(Leximax ordering and leximax optimal solutions)*
Let x and y denote two vectors in $\mathbb{R}^m$. Let $x^{\downarrow}$ and $y^{\downarrow}$ denote these same vectors where each element has been rearranged in a non-increasing order. According to the leximax ordering:

- *the vector x leximax-dominates y (written $x \prec_{leximax} y$) if and only if $\exists i \in \{1, ..., m\}$ such that: $\forall j \in \{1, ..., i-1\}, x_j^{\downarrow} = y_j^{\downarrow}$ and $x_i^{\downarrow} < y_i^{\downarrow}$,*
- *x and y are indifferent (written $x \sim_{leximax} y$) if and only if $x^{\downarrow} = y^{\downarrow}$.*

We write $x \preceq_{leximax} y$ the case where $x \prec_{leximax} y$ or $x \sim_{leximax} y$.
Let S be a set of vectors in $\mathbb{R}^m$ representing the feasible solutions of an optimization problem, then x is leximax-optimal *if $\forall y \in S, x \preceq_{leximax} y$.*

In other terms, if we denote by $\leq_{lex}$ the lexicographic ordering, then:

$$x^{\downarrow} \leq_{lex} y^{\downarrow} \Leftrightarrow x \preceq_{leximax} y.$$

In this paper, we consider the capacitated vehicle routing problem (CVRP) Toth and Vigo (2002) in which a set of customers should be delivered by a set of m vehicles from a depot, such that the quantity delivered by each vehicle is not greater than its capacity Q. For the purpose of notation in the remaining of the paper, we consider that the weight of each arc represents the duration as well as the cost of traveling through this arc. The cost of a route is then the sum of the weights of all arcs traversed by this route. The duration of a route is the sum of the weights of all arcs traversed and the service durations at all customers visited by this route. The duration of each route should be less than a given maximum value denoted by L.

We consider the problem of finding the set of solutions to the CVRP that offer the best trade-off between routing costs and load balancing. The leximax ordering over route durations is taken to compare solutions on workload balancing aspects. Although the leximax ordering cannot be explicitly called an objective function because it does not define an objective value for a given solution, we consider the problem as a bi-objective optimization problem.

We call this problem the *leximax*-VRP. Let s be a solution to the *leximax*-VRP. We denote by c_s the cost of a solution s and $l^s = (l_1^s, ..., l_m^s)$ the vector of route durations for this solution. Pareto optimal solutions to the *leximax*-VRP can be defined as follows:

Definition 2. *(Dominance and Pareto optimality in the leximax-VRP)*
Let s and s' represent two solutions of the leximax-VRP. A solution s dominates *a solution s' iff: $c_s \leq c_{s'}$ and $l^s \preceq_{leximax} l^{s'}$ and either $c_s < c_{s'}$ or $l^s \prec_{leximax} l^{s'}$.*
s is a Pareto optimal *solution iff no other solution dominates s.*

Definition 2 is illustrated in Example 2.

We consider a leximax-VRP with 4 vehicles and the 6 solutions described in Table 1. A graphical visualization of these solutions is given in two dimensions (Solution cost and maximum route duration) on Figure 1. The blue discs represent the Pareto optimal solutions. The minimum cost solution is s1, and the most balanced is s5. Solution 6 is dominated by s3, s4, and s5. Note that this solution has the minimum range. Solution s2 is not dominated by s1 although it has a greater

Example 2.

Solution	Cost	Sorted route durations
s1	15	14, 11, 3, 2
s2	17	14, 9, 5, 4
s3	20	12, 11, 5, 5
s4	22	12, 11, 6, 6
s5	24	10, 10, 10, 9
s6	33	12, 12, 12, 12

Table 1: Solutions representation according to their cost and routes duration.

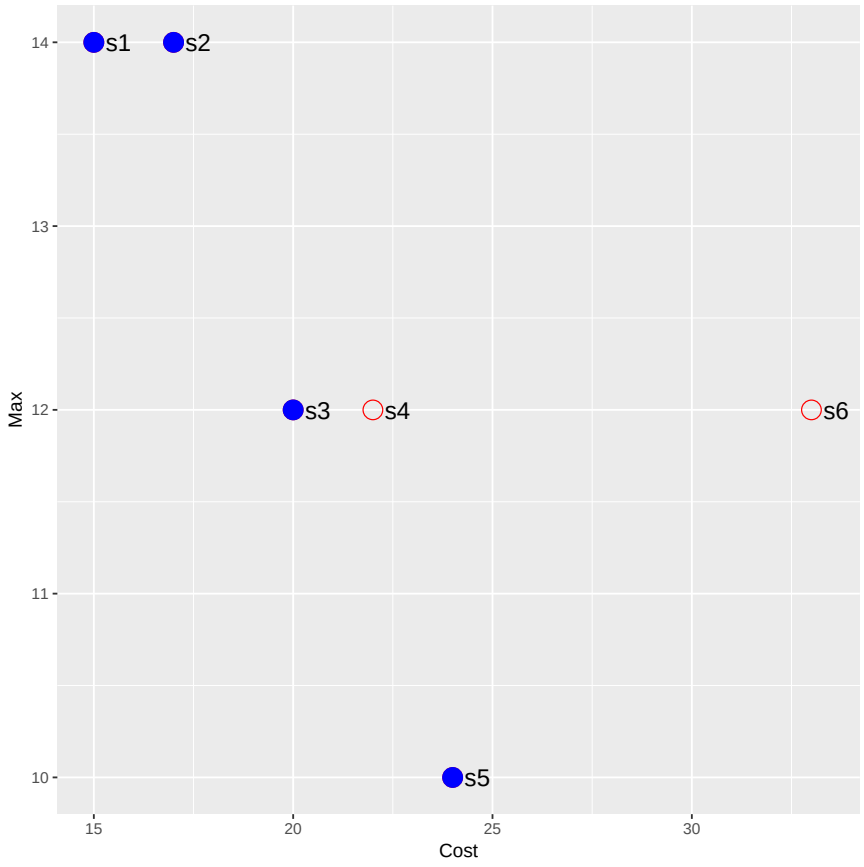


Figure 1: Two-dimension solutions visualization according to the duration of the longest route in each solution (Max) and the solution cost.

cost and the same maximum route duration as s2. Indeed, it offers a better balance than s1 between the second, third and fourth routes of the solution. Conversely, s4 is dominated by s3 because it is worse than s3 for cost and because the vector of route durations of s3 leximax-dominates the vector of route durations of s4.

The goal of our approach is to compute the complete set of Pareto optimal solutions to the *leximax*-VRP (called Pareto set). As the problem is NP-hard, we try to approximate this set with a bi-objective metaheuristic.

In the context of optimization algorithms, one complicating factor is that the lexicographic minimax approach does not measure a solution's quality with a single scalar value. Instead, it is based on the binary relation $\preceq_{leximax}$, which allows to compare two solutions and determine if one is more balanced than the other. This comparison involves sorting and comparing two vectors, rather than simply comparing two numbers. However, we believe that this binary relation defines a dominance relation that can easily be integrated in a metaheuristic.

3. Solution method

To solve the *leximax*-VRP, we propose to integrate the lexicographic minimax approach in the multi-directional local search (MDLS) framework introduced by Tricoire (2012). We first recall the principles of the generic framework in Section 3.1. In Section 3.2, we describe the LNS operators that have been used to produce solutions to the *leximax*-VRP in this framework.

3.1. Multi-directional local search

MDLS is a multi-objective optimization framework (Tricoire, 2012) which generalizes the concept of local search to multiple objectives. It is an iterative method, where the number of iterations (or CPU budget) is the only problem-independent parameter. MDLS considers an archive F of solutions which is maintained and updated through the solution process. At each iteration, three steps are performed:

1. A solution x is randomly selected from F, each solution having equal probability.
2. For each objective, single-objective local search is performed on x.
3. F is updated with solutions obtained by single-objective local search.

In the following, we call *non-dominated set* any set of solutions such that no solution from this set dominates another solution from this set. MDLS maintains F as a non-dominated set by performing *non-dominated union* operations. The *non-dominated union* of two sets performs the union of these two sets while filtering out the elements that are dominated by at least one element of these two sets. Non-dominated union is defined for any multi-objective setting and is not specific to the *leximax*-VRP. We note it $\cup_{\preceq}$ and define it as follows: $A \cup_{\preceq} B = \{x \in A \cup B \mid \nexists y \in A \cup B, y \neq x \wedge y \preceq x\}$, where $y \preceq x$ means that y dominates x for a given multi-objective optimization problem.

In order to apply MDLS to the *leximax*-VRP, we need to define local search for both the cost and leximax objective functions. For that purpose we develop a parametric large neighborhood search algorithm, called LNS_k, where parameter k determines the evaluation function being used, i.e. the objective being optimized: LNS_1 aims at improving cost while LNS_2 aims at minimizing the leximax value. LNS is a iterative metaheuristic that iteratively *destroys* and *repairs* an incumbent solution (Shaw, 1998); it has been successfully applied to a variety of routing problems, see e.g. Pisinger and Ropke (2007). The LNS algorithm itself is detailed in section 3.2. LNS has

become one of the most successful paradigms for solving various transportation and scheduling problems (see Pisinger and Ropke (2019)).

Algorithm 1 outlines our LNS-based MDLS algorithm, taking as parameters a non-dominated archive F and a CPU time budget *timeLimit*.

Algorithm 1 $MDLS(F, timeLimit)$

```
pre-condition: F is a non-dominated set
repeat
    x ← select_a_solution(F)
    F ← ∅
    for k ∈ {1, 2} do
        F ← F ∪_⪯ LNS_k(x)
    end for
until timeLimit is reached
return F
```

The archive F is initialized with the solution of the *cheapest insertion heuristic* adapted from Pisinger and Ropke (2007). In the first MDLS iterations, solutions may not be feasible: with a limited number of vehicles, we may end up with a solution where some unvisited customers cannot be inserted into any route, due to bin packing (capacity) constraints. Since feasibility cannot be guaranteed, we accept infeasible solutions to begin with. The dominance rule needs to be adapted to tackle infeasibility: if a solution x^1 has less unvisited customers than another solution x^2, then x^1 always dominates x^2.

3.2. Local search components

In our implementation of MDLS, we consider that local search consists of one Large Neighborhood Search (LNS) iteration Ropke and Pisinger (2006). Several ruin and recreate operators are defined for each objective. Hence, at each iteration, for each objective (i) a ruin and a recreate operator are randomly selected in the set of operators for that objective and (ii) a new solution is produced using the selected operators. According to the LNS framework Pisinger and Ropke (2019), the *ruin quantity*, used in the destroy operator, is randomly selected within some bounds (at least one customer and at most 30% of the number of customers in our experimentations). All customers removed from the current solution by the *removal operator* are placed in a request bank denoted $\mathcal{B}$. The goal of the *repair operator* is to re-insert all customers $i \in \mathcal{B}$ in the partial solution resulting from the removal operator.

3.2.1. Cost oriented ruin and recreate operators

We first implement some classical LNS removal operators and repair operators defined for the VRP with the single objective of cost minimization. The removal operators that we use are *random removal, worst removal, related removal* and *route removal*. The recreate operators are the *cheapest insertion heuristic* and the *k-regret heuristic* for $k = 2, 3, 4$. The operators are described in many articles, e.g., Pisinger and Ropke (2007). We call them *cost-oriented operators*.

3.2.2. Leximax ruin operators

A major contribution of this work is to introduce *leximax operators*. They constitute rather natural extensions of the relevant classical operators to the lexicographic minimax approach. The

ruin operators include the *random removal* and the *related removal* as well as the following two operators:

- *worst max removal:* this operator removes, from the longest route, the customer that decreases the most the duration of this route. This is repeated until the number of removed customers is equal to the ruin quantity.
- *longest route removal:* this operator removes all customers from the longest route. This is repeated until the number of removed customers is greater than or equal to the ruin quantity. Several routes may be destroyed this way.

3.2.3. Leximax recreate operators

Two sets of recreate operators have been designed to guide the search towards leximax-optimal solutions:

- The *leximax cheapest insertion* and *leximax k-regret* extend the classical cheapest insertion heuristic and the k-regret heuristic according to the lexicographic minimax approach.
- The *min-max cheapest insertion* and *min-max k-regret*, which are straightforward simple extensions of cheapest insertion and k-regret using the duration of the longest route to guide the search.

The *leximax cheapest insertion* is a rather natural extension of the cheapest insertion heuristic. At each iteration, a customer $i \in \mathcal{B}$ is selected and inserted at its best position in a route r. The resulting solution must dominate (according to the leximax ordering) any other solution resulting from the insertion of i in any other route. It must also dominate any other solution resulting from the insertion of customer $i' \in \mathcal{B}$, $i' \neq i$ in the current solution.

The *leximax k-regret operator* works as follows: for a given vertex $i \in \mathcal{B}$, we note $d^i_1, ..., d^i_m$ the m vectors of route durations that result from the insertion of i at its best insertion in routes $1, ..., m$.

Let σ be a permutation of route indices such that $d^i_{\sigma(1)} \preceq_{leximax} ... \preceq_{leximax} d^i_{\sigma(m)}$. We define the leximax k-regret of customer i as the vector:

$$\Delta^i_k = \sum_{j=2}^{k} (d^{i\downarrow}_{\sigma(k)} - d^{i\downarrow}_{\sigma(1)}).$$

At each iteration, the vertex i with the greatest regret according to the lexicographic non-decreasing ordering (i.e. such that $\forall i' \in \mathcal{B}$, $\Delta^i_k \geq_{lex} \Delta^{i'}_k$) is inserted in the solution at its best position in the route that yields the best solution w.r.t. the leximax order (i.e., the solution with route durations $d^i_{\sigma(1)}$).

The leximax-based repair operators are computationally expensive: comparing two insertions means sorting two vectors of size m and comparing them. Sorting a vector of size m can be done in $\mathcal{O}(m(\log m))$, but in fact there is never a need to sort the vector entirely, since only one element is modified for a given insertion. Therefore it is sufficient to remove the route duration being modified and insert the modified value, both operations being in $\mathcal{O}(\log m)$. Then both vectors still need to be compared, which in the worst case is in $\mathcal{O}(m)$. Therefore we can approximate the worst-case time complexity of any given comparison of two insertions to $\mathcal{O}(m)$.

In contrast, comparing two insertions for the cost objective is in $\mathcal{O}(1)$. This means that we can expect to spend a majority of the CPU budget on the leximax-based repair operators. In order to determine whether it is worth it, we also develop operators that aim at minimizing the leximax, but which operate in constant time: the *min-max cheapest insertion* and *min-max k-regret* are simplified versions of the leximax operators, which use only the duration of the longest route to guide the search, as well as the solution cost increase to break ties. The objective of introducing these operators is to assess the efficiency of these faster operators to guide the search towards lexicographic minimax solutions.

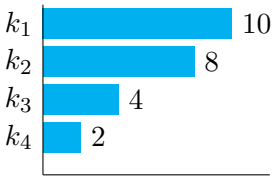
Example 3. *Leximax 2-regret operator*
We illustrate the proposed operator on the example represented on Figures 2 and 3. Let us consider an incomplete current solution with four routes $k_1, ..., k_4$ with durations 10,8,4 and 2, respectively, as represented on Figure 2a.

Let i and j be two un-routed customers. The four charts of Figure 2b show the route durations that result from the insertion of vertex i at its best position in each of the four routes of the solution. Hence, each chart represents a partial solution. On Figure 2c, these four solutions are sorted according to the leximax operator: they are displayed with their routes sorted on a decreasing order so that the leximax order corresponds to the lexicographic non-decreasing order. On Figure 2d, the regret value is calculated as the difference between the vector $d^{i\downarrow}_{\sigma(2)}$ that represents the second best insertion and the vector $d^{i\downarrow}_{\sigma(1)}$ that represents the best insertion according to the leximax order.

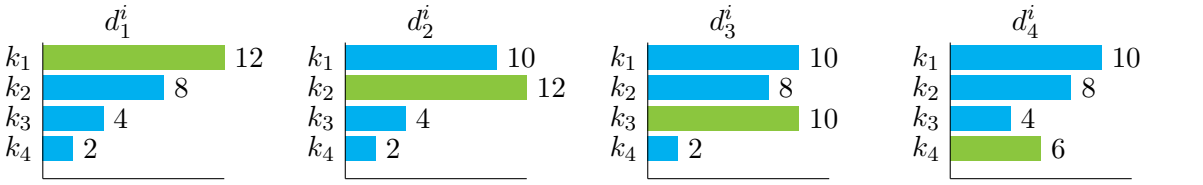
The calculation of the regret of vertex j is illustrated on Figure 3. This regret is then compared to the regret of vertex i (Figure 3c) using the lexicographic decreasing order. Since j has the greatest regret, it is selected for insertion. According to the order of Figure 3b, j is inserted at its best position in route k_1.

Note that on this example, the leximax cheapest insertion operator would have compared $d^{i\downarrow}_{\sigma(1)}$ and $d^{j\downarrow}_{\sigma(1)}$ using a non increasing lexicographic order. Its conclusion would have been to insert vertex i at its best position in route k_4. The leximax 2-regret operator anticipates that the second best insertion of j yields a longest route that is significantly higher than the one obtained for its best insertion. This is not the case for vertex i, therefore, leximax 2-regret recommends to insert j before i.

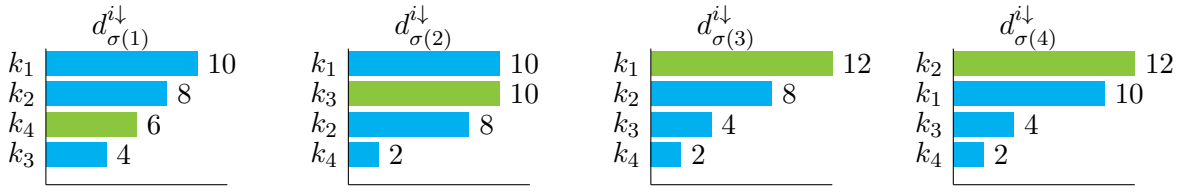
(a) Route durations of the current solution (with vertices i and j to be inserted):



(b) Evaluation of the insertion of vertex i in each route:



(c) Order with respect to $\prec_{leximax}$:



(d) Caculate 2-regret:

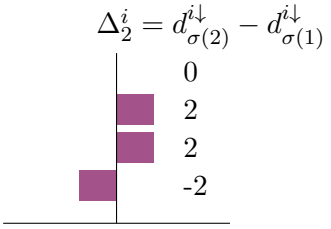


Figure 2: Example lexi-2-regret: insertion of a vertex i

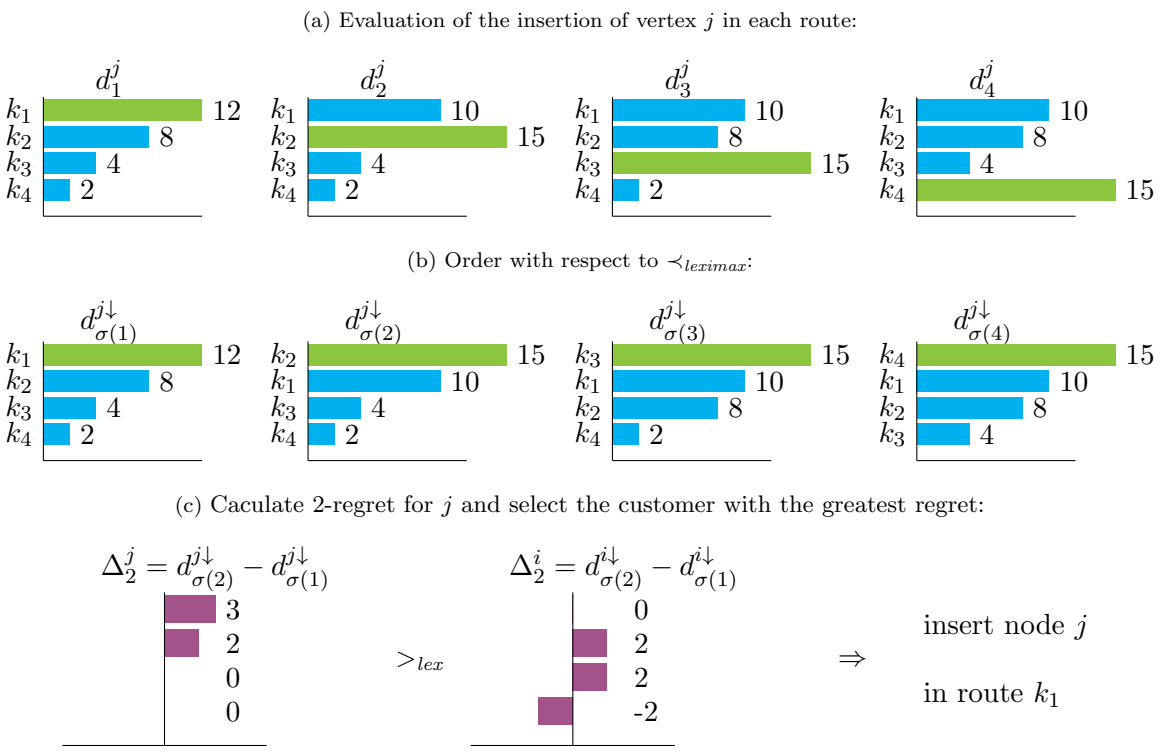


Figure 3: Example lexi-2-regret: insertion of a vertex j and selection of the request with the highest regret

4. Computational experiments

The designed MDLS is evaluated on the Christofides CVRP instances (Eilon et al., 1971; Christofides et al., 1979), which have been traditionally used to benchmark the VRP with route balancing (Jozefowiez et al., 2009; Lacomme et al., 2015). We ignore the route duration limit that is imposed on some instances since all solutions satisfying this limit can be easily identified in the final set produced by MDLS. For each instance, the fleet size is set as the number of vehicles in the best known solution reported by CVRPLIB (2018).

The main characteristics of the 14 instances are presented in Table 2.

For all instances and solution methods, 10 runs were performed on an Intel(R) Xeon(R) X5675 3.07GHz processor.

4.1. MDLS configurations comparison

We test three MDLS configurations, called *leximax*, *max* and *all*, respectively. The three configurations include all removal operators as well as all cost oriented operators. The list of operators used to optimize each objective in MDLS and for each configuration is detailed in Table 3.

The *max* configuration includes Min-max cheapest insertion and Min-max k-regret, which are the simple and fast extensions of the traditional LNS recreate operators based on cost. The *leximax* configuration includes the leximax cheapest and leximax k-regret operators. The *all* configuration includes all recreate operators. Comparing the three configurations we aim at evaluating if very simple search strategies may be applied to efficiently find solutions or if the more elaborate operators are needed.

Instance	# customers	BKS # vehicles	BKS cost	Maximum duration	Service time
vrpnc1	50	5	524.61	$+\infty$	0
vrpnc2	75	10	835.26	$+\infty$	0
vrpnc3	100	8	826.14	$+\infty$	0
vrpnc4	150	12	1028.14	$+\infty$	0
vrpnc5	199	17	1291.45	$+\infty$	0
vrpnc6	50	6	555.43	200	10
vrpnc7	75	11	909.68	160	10
vrpnc8	100	9	865.94	230	10
vrpnc9	150	14	1162.55	200	10
vrpnc10	199	18	1395.85	200	10
vrpnc11	120	7	1042.11	$+\infty$	0
vrpnc12	100	10	819.56	$+\infty$	0
vrpnc13	120	11	1541.14	720	50
vrpnc14	100	11	866.37	1040	90

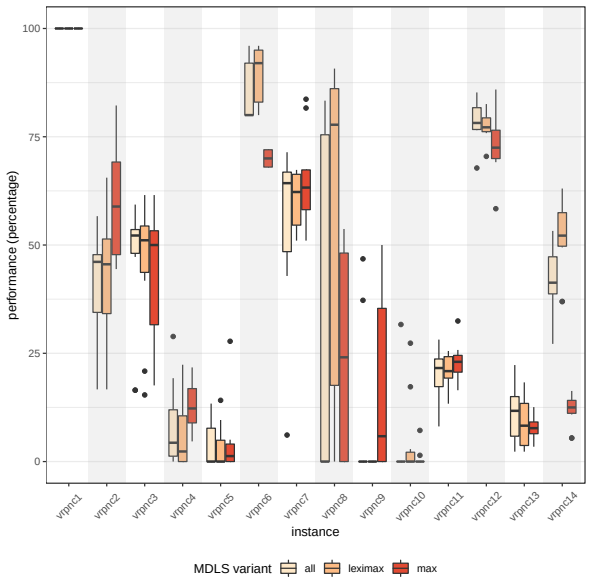
Table 2: Characteristics of the CVRP instances that have been adapted for our experiments. The columns specify the number of customers in each instance, the number of vehicles in the best known solutions for these instances (proven optimal for instances 1-5, 11, 12), the cost of best known solutions, the maximum allowed duration for each route, and the duration of service at customer locations.
For the leximax-VRP: the number of vehicles in the best known CVRP solution is taken as number of available vehicles; maximum duration constraints are not considered.

Operator	Objective		Configuration		
			max	*leximax*	*all*
Random removal	❶	❷	✔	✔	✔
Worst removal	❶		✔	✔	✔
Related removal	❶	❷	✔	✔	✔
Route removal	❶		✔	✔	✔
Worst max removal		❷	✔	✔	✔
Longest route removal		❷	✔	✔	✔
Cheapest insertion	❶		✔	✔	✔
k-regret ($k = 2, 3, 4$)	❶		✔	✔	✔
Leximax cheapest insertion		❷	✗	✔	✔
Leximax k-regret ($k = 2, 3, 4$)		❷	✗	✔	✔
Min-max cheapest insertion		❷	✔	✗	✔
Min-max k-regret ($k = 2, 3, 4$)		❷	✔	✗	✔

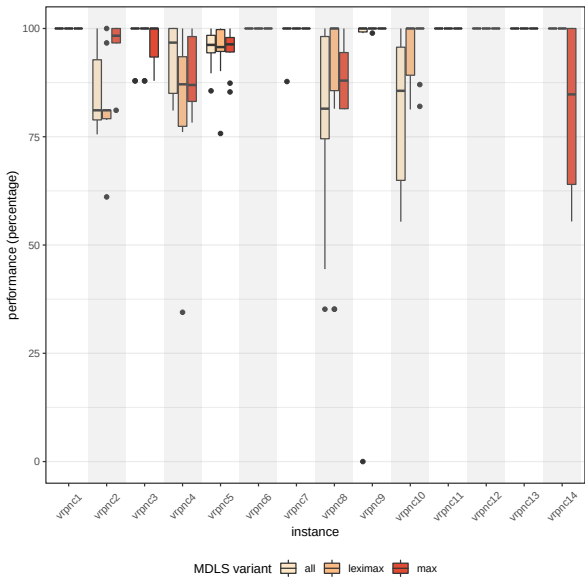
Table 3: Summary of operators and MDLS configurations

4.1.1. Quality of the Pareto front approximation

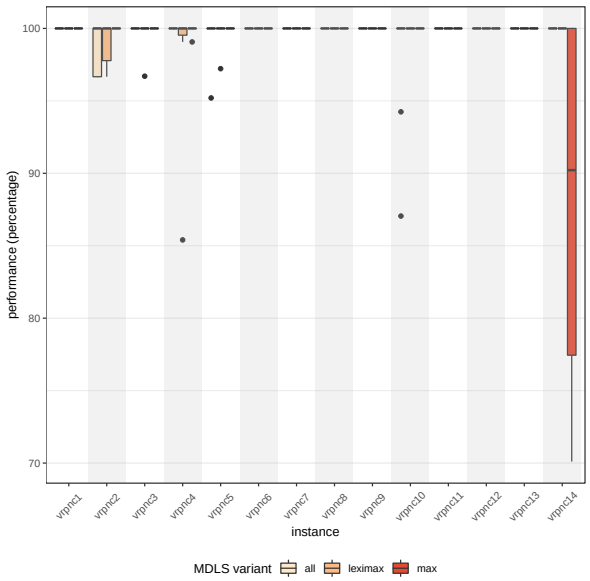
As presented in Section 4.2, the configurations are tested with runtimes of 1, 10, 30 and 60 minutes. For each instance, a *reference set* is constructed by taking the non-dominated union of the sets returned by each run for each configuration and each tested run time ($10 \times 3 \times 4$ sets in total). We evaluate the quality of our approximation of the Pareto set, first by looking at the percentage of solutions from the reference set found for each run in the 60-minutes benchmarks. These results are represented on Figure 4. This figure shows four box-and-whisker diagrams (boxplots) which represent the percentage of reference solutions found as well as those that are within a 1%, 2% or 3% distance of a solution from the reference set. A solution x_1 is said to be within a α% distance of another solution x_2 if, when the cost and all route durations of x_2 are multiplied by $1 + \frac{\alpha}{100}$, then x_1 dominates this transformed solution. For each instance and each run we look at the proportion of the solutions in the reference front for which the algorithm returned a solution within the specified distance. For each MDLS configuration and each instance, the boxplot represents the variability of the results returned by the 10 runs.



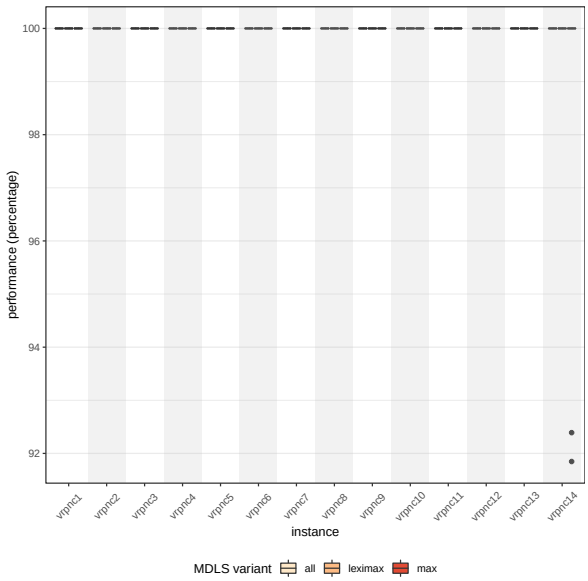
(a) % of the reference set found by each MDLS configuration



(b) % of the reference set within 1% distance from a found solution



(c) % of the reference set within 2% distance from a found solution



(d) % of the reference set within 3% distance from a found solution

Figure 4: Proportion of solutions in the reference set found within a 0%, 1%, 2%, 3% distance (run time: 60 minutes)

On boxplot 4a, the three tested configurations of MDLS consistently find more than 50% of the reference set for only three instances. However, for most instances and most methods, the returned set is actually relatively close to the reference set, as shown in Figures 4b to 4d.

Comparing the different MDLS configurations, it can be observed that, although the *max* configuration seems slightly better for some instances on boxplot 4a, it experiences some difficulties on instance vrpnc14. Indeed, boxplot 4c shows that large proportions of the reference set are more than 2% away from any solution returned by *max* on this instance on most runs of the algorithm.

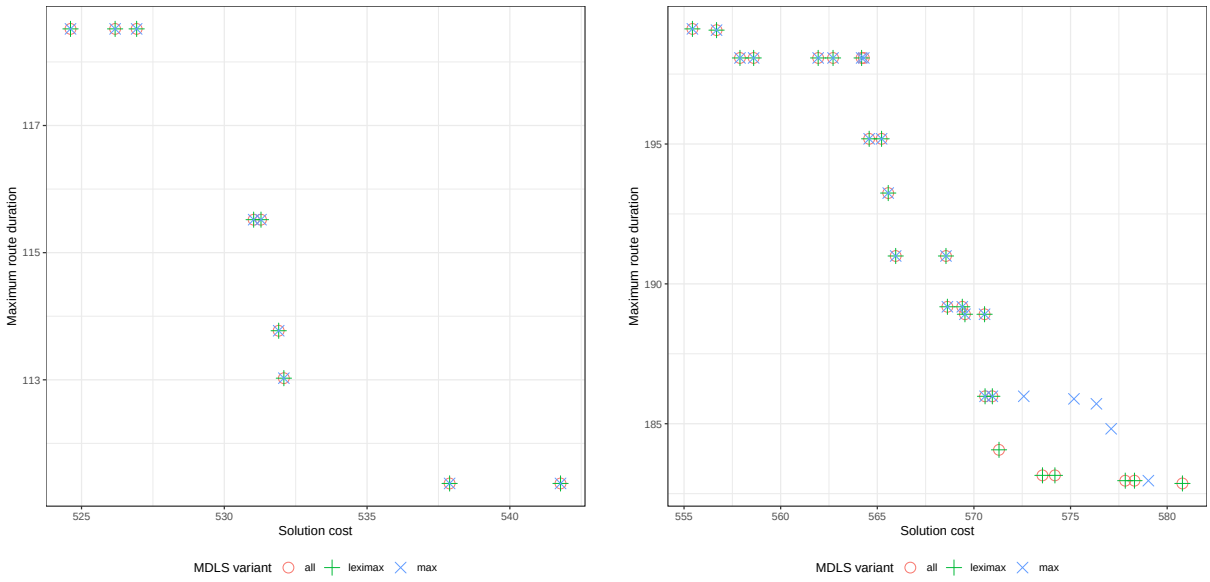
The synthesis proposed by Figure 4 is further analyzed in graphical representations of the solution set obtained by the various configurations for several instances, as depicted on Figures 5 and 6. Here, we compare the union of the sets returned by the 10 runs for the three MDLS configurations after 60 minutes of computation. The horizontal axis represents the solution cost, whereas the vertical axis represents the duration of the longest route. Note that in theory, two solutions may have similar cost and Max and different route durations – but we did not observe such cases and believe they are quite exceptional.

Figure 5 represents the union of the MDLS-produced sets for four instances (vrpnc1, vrpnc6, vrpnc7 and vrpnc12) for which the three MDLS configurations give the most satisfactory results (from Figure 4). Figure 6 represents the union of the MDLS-produced sets for four instances (vrpnc2, vrpnc8, vrpnc10 and vrpnc14), for which the three MDLS configurations seem further from the reference front.

From these figures we can see that, for most cases, all configurations provide very similar approximations of the Pareto set. In particular, the simple construction heuristics of the max configuration are generally sufficient to reach a good enough set. Nevertheless, Figures 5b, 6b and 6c suggest that leximax-based heuristics, integrated in the *leximax* and *all* configurations, slightly help to find the most balanced solutions on some instances. Figure 6c shows that none of the configurations really achieves a good convergence for instance vprnc10, which is the largest instance.

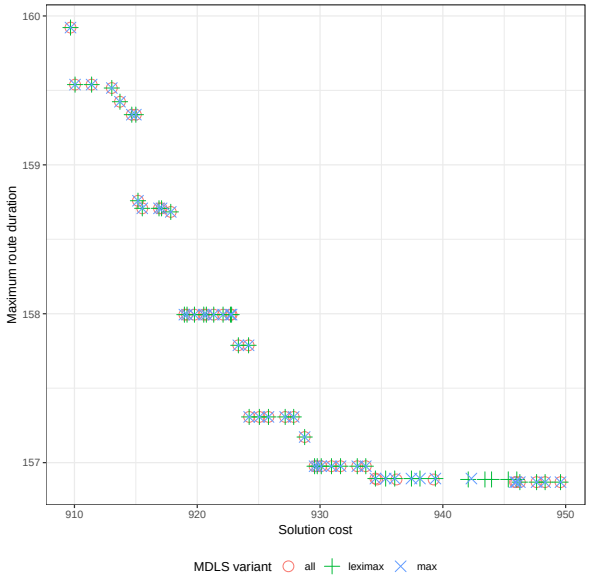
4.1.2. Instances with larger fleet size

To further investigate the comparison between the three MDLS configurations, we now use instances which yield more opportunities for balancing the workload. To that end, we consider the same instances as previously but using $m^* + 1$ vehicles, where m^* is the number of vehicles considered previously, i.e. the number of vehicles in the best known solution for the cost objective. This setting captures the fact that certain workdays have a lower activity, thus making workload balancing a greater challenge.

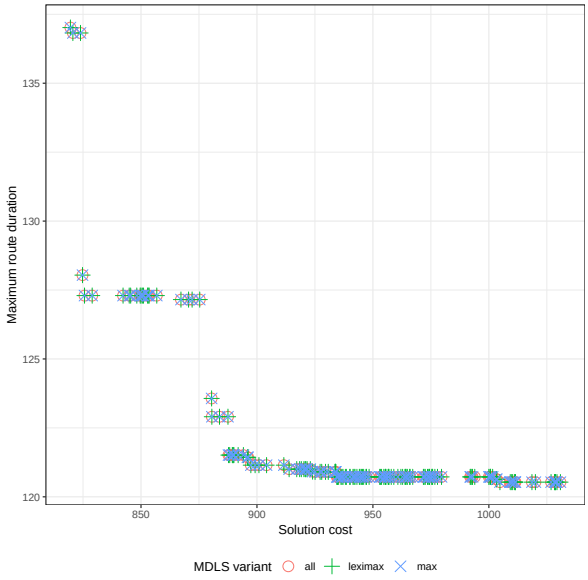


(a) Instance vrpnc1

(b) Instance vrpnc6

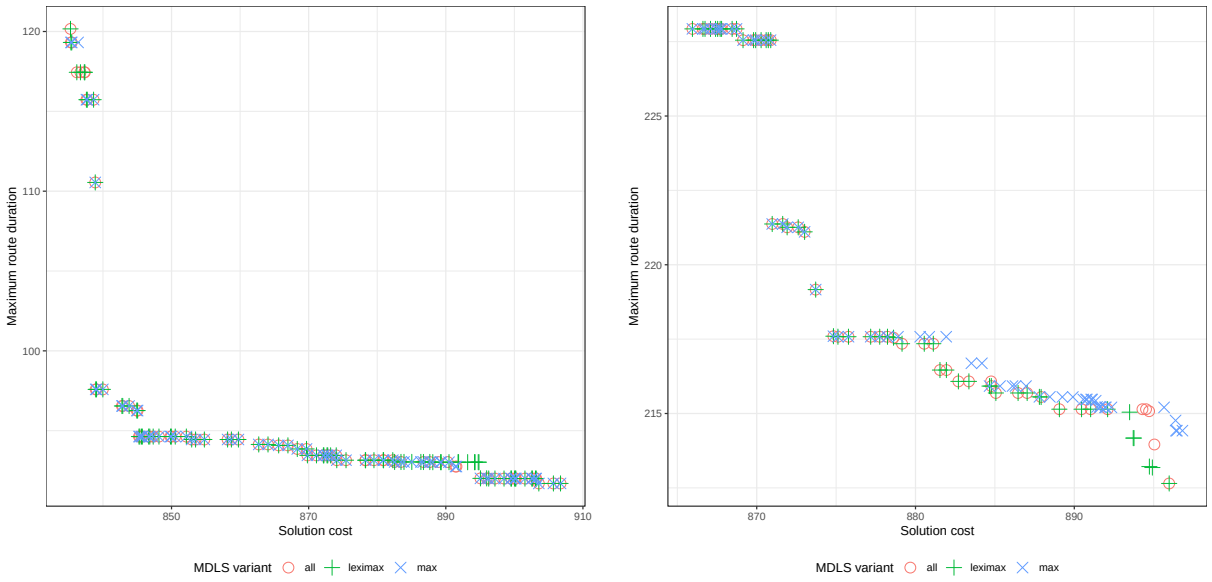


(c) Instance vrpnc7



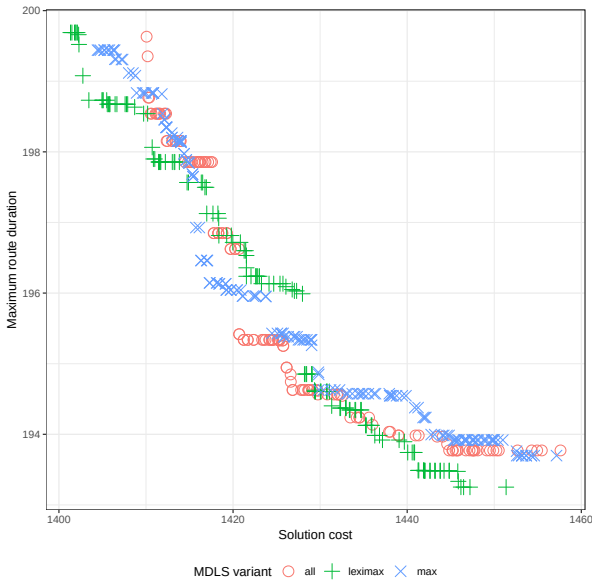
(d) Instance vrpnc12

Figure 5: Union of solution sets for the vrpnc1, vrpnc6, vrpnc7 and vrpnc12 instances.

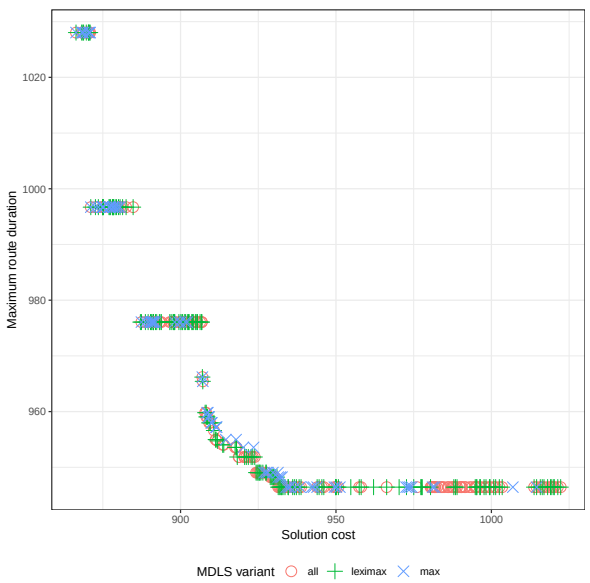


(a) Instance vrpnc2

(b) Instance vrpnc8

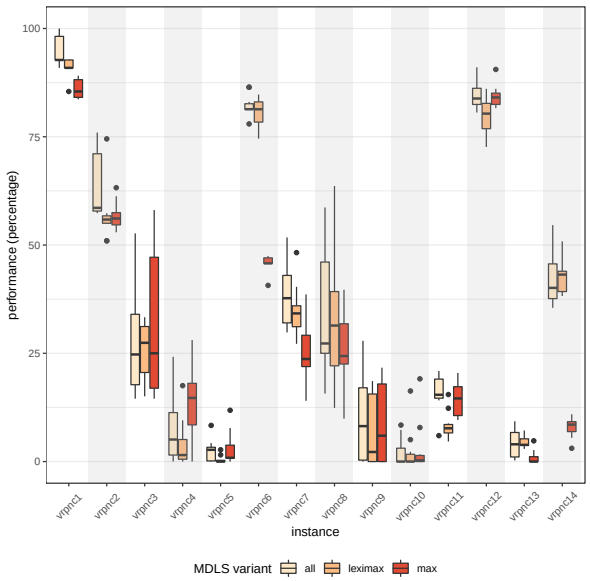


(c) Instance vrpnc10

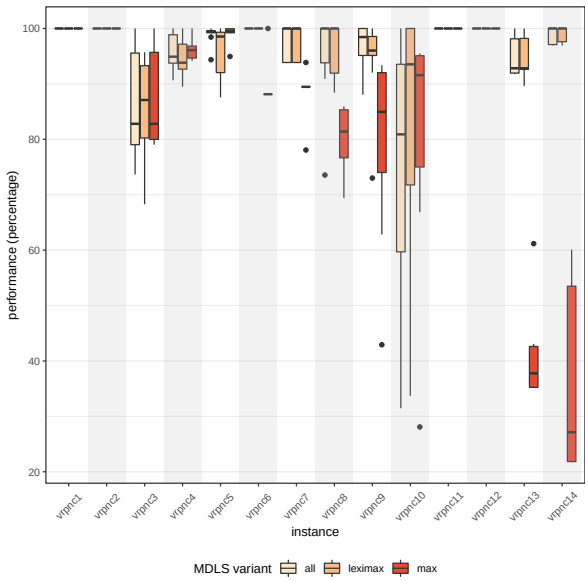


(d) Instance vrpnc14

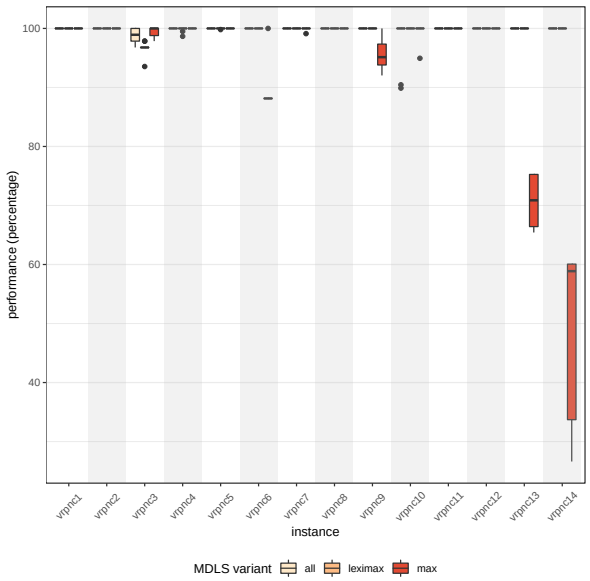
Figure 6: Union of solution sets for the vrpnc2, vrpnc8, vrpnc10 and vrpnc14 instances.



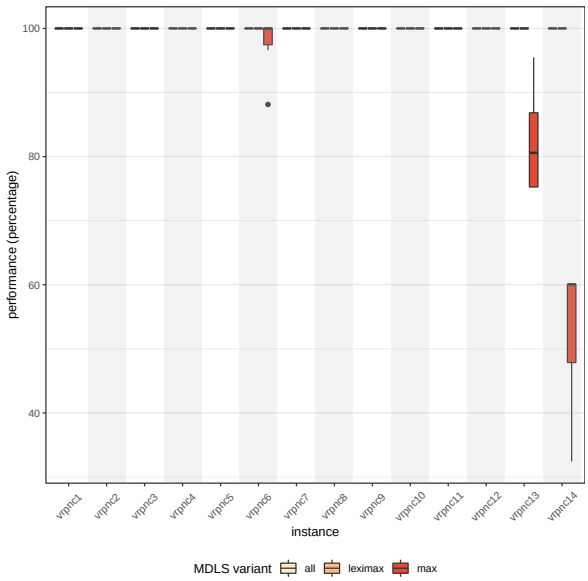
(a) % of the reference set found by each MDLS configuration



(b) % of the reference set within 1% distance from a found solution



(c) % of the reference set within 2% distance from a found solution

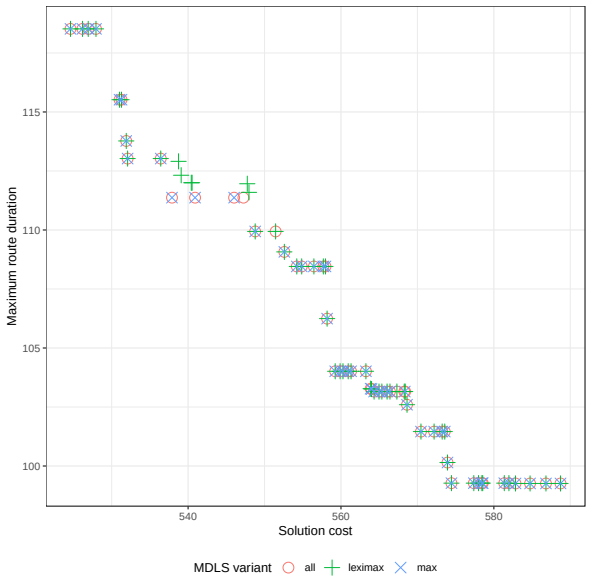


(d) % of the reference set within 3% distance from a found solution

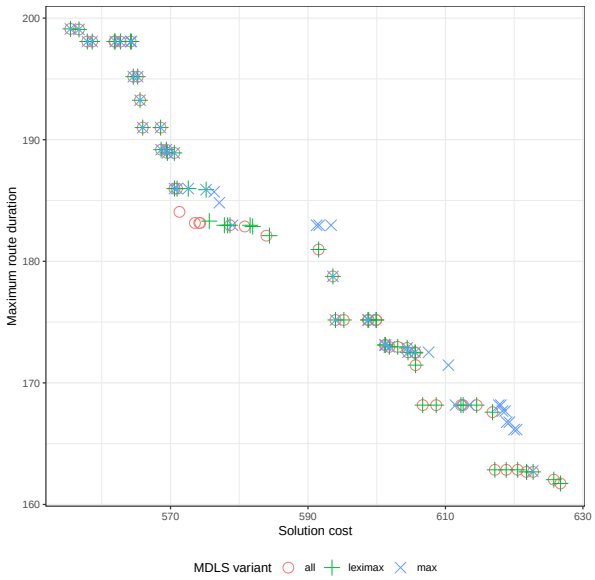
Figure 7: Proportion of solutions in the reference set found within a 0%, 1%, 2%, 3% distance when $m = m^\star + 1$ (run time: 60 minutes)

First, the four boxplots of Figure 7 show that the instances become significantly harder with an additional vehicle. For example, on Figure 4a, all methods are able to get the reference set on all runs for instance vrpnc1 with $m = m^\star$. This is not the case anymore on Figure 7a. Second, it becomes more obvious that the *max* configuration can be less efficient than the others on some instances.

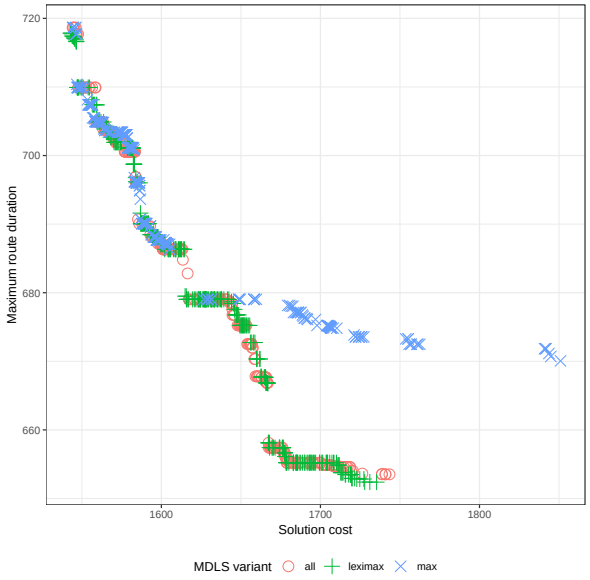
On Figures 8c and 8d, further insights are proposed by plotting the unions of the solution



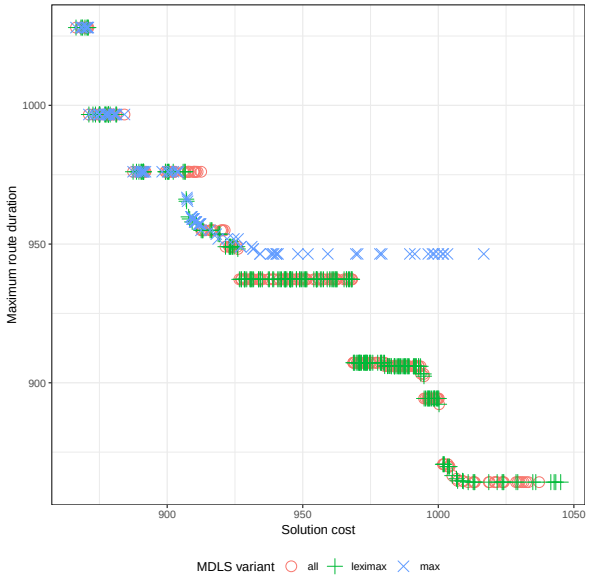
(a) Instance vrpnc1



(b) Instance vrpnc6



(c) Instance vrpnc13



(d) Instance vrpnc14

Figure 8: Union of solution sets for four instances with one additional vehicle.

sets returned by all runs for several instances. It can be observed that the *max* configuration is outperformed by the others mainly on the most balanced solutions. This indicates that in some cases, the *min-max cheapest insertion* and *min-max k-regret* heuristics may not be able to spread the load over drivers as efficiently as the leximax based heuristics.

4.1.3. Number of iterations

To evaluate the impact of the higher complexity of the leximax recreate operators, we compare the number of iterations performed by MDLS with the various configurations in the same runtime. These comparisons are performed, taking the minimum number of vehicles for each instance.

On Figure 9, the 14 instances are placed on the horizontal axis. The vertical axis represents the number of iterations performed after 1 minute. Quite logically, integrating the leximax regret and leximax best insertion has a significant impact on the average duration of each iteration, up to almost twice as many iterations in the case of instance vrpnc2.

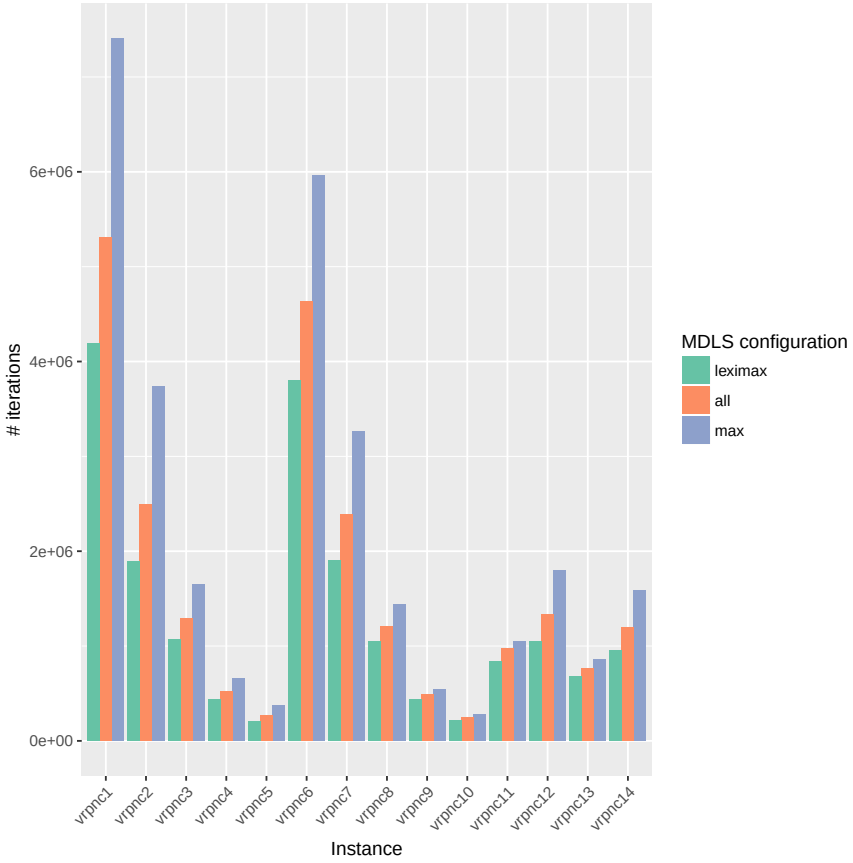


Figure 9: Number of MDLS iterations after one minute, for each configuration on each instance.

Figure 9 completes the analysis of Section 4.1.1 by illustrating the following observation: The *max* configuration implements heuristics that are less complex. As a result, more MDLS iterations can be performed within its time limit. This partly explains why all configurations show the same performance for several instances. However, for some instances such as vrpnc6 on Figure 5b, using the *leximax* heuristics is important to obtain most trade-off solutions, even though the *max*

configurations performs almost 50% more iterations than *leximax*.

4.1.4. Experiments on small instances

To complete the previous experiments, we evaluate the performance of the proposed MDLS on small instances for which all solutions have been enumerated in order to extract the exact *leximax*-VRP Pareto front. We created 14 small instances, each having ten customers and five vehicles. These have been created by taking the ten first customers of instances vrpnc1 to vrpnc14 and are called vrpnc1|10 to vrpnc14|10. Ten runs of each MDLS configurations have been performed for each instance with a time limit of one second. For each instance and each configuration, we keep the reference front, which is the union of the non-dominated solution sets of all runs. Instances and results are presented in Table 4. The first column indicates the original instance. For each instance, the vehicles capacities (column 2) have been manually adjusted such that feasible solutions can be found with five vehicles and the number of non-dominated solutions is large enough. The third column presents the number of non dominated solutions of the *leximax*-VRP that have been calculated by enumerating all feasible solutions to the problem and filtering the dominated ones. In columns 4 to 6 we indicate the number of solutions of the Pareto front that have not been found by the three MDLS configurations. Over all instances, the *max* configurations misses five solutions, the *leximax* configuration misses two solutions and configuration *all* misses only one solution.

This experiment confirms the previous conclusions on larger instances: the three MDLS configurations provide results that are equivalent most of the time. Nevertheless, for some instances, some trade-off solutions are harder to find for the *max* configuration. In this case, the leximax based operators are better to guide the search towards good solutions.

Instance	Vehicle capacity	# of sol. in the Pareto front	# of sol. **not** found by configuration		
			leximax	all	max
vrpnc1|10	40	5	0	0	0
vrpnc2|10	60	6	0	0	0
vrpnc3|10	36	5	0	0	0
vrpnc4|10	38	7	0	0	0
vrpnc5|10	40	9	0	0	0
vrpnc6|10	50	8	0	0	0
vrpnc7|10	55	9	0	0	0
vrpnc8|10	40	7	0	0	0
vrpnc9|10	39	7	0	0	0
vrpnc10|10	35	4	0	0	0
vrpnc11|10	40	9	0	0	3
vrpnc12|10	40	13	0	0	0
vrpnc13|10	45	15	1	0	1
vrpnc14|10	50	16	1	1	1

Table 4: MDLS results on small instances for which the exact set of non dominated solutions can be enumerated

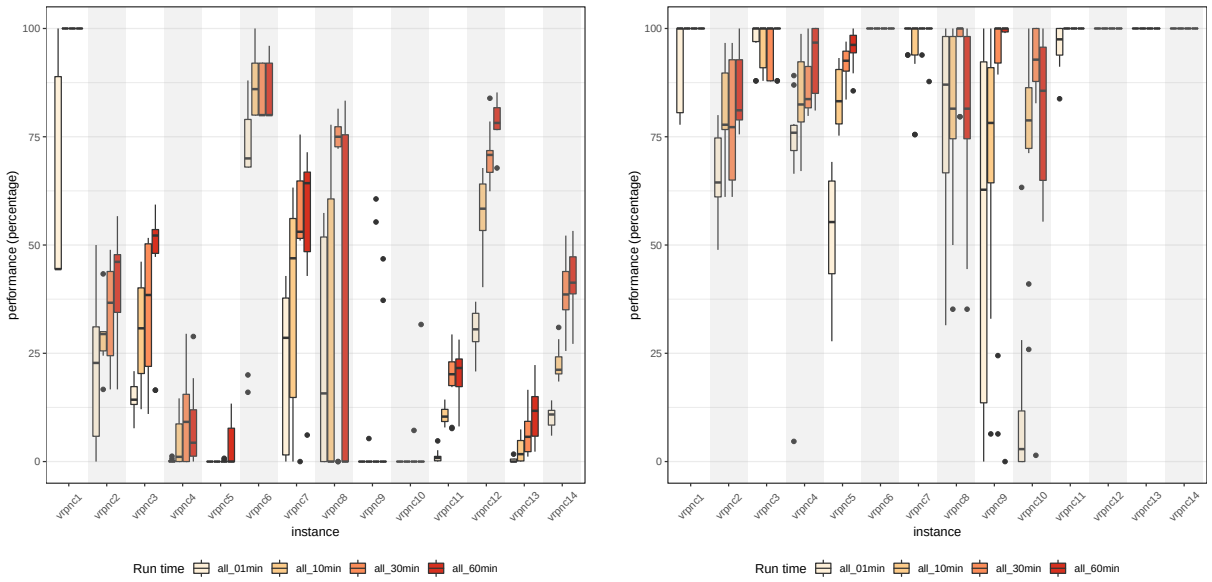
4.1.5. Discussion

As a conclusion, from these experiments, the combination of leximax and min-max operators in configuration *all* offer a good trade-off for solving the *leximax*-VRP. The *leximax* recreate operators

are slower, but given the same CPU budget they increase the quality of the solution sets produced by MDLS.

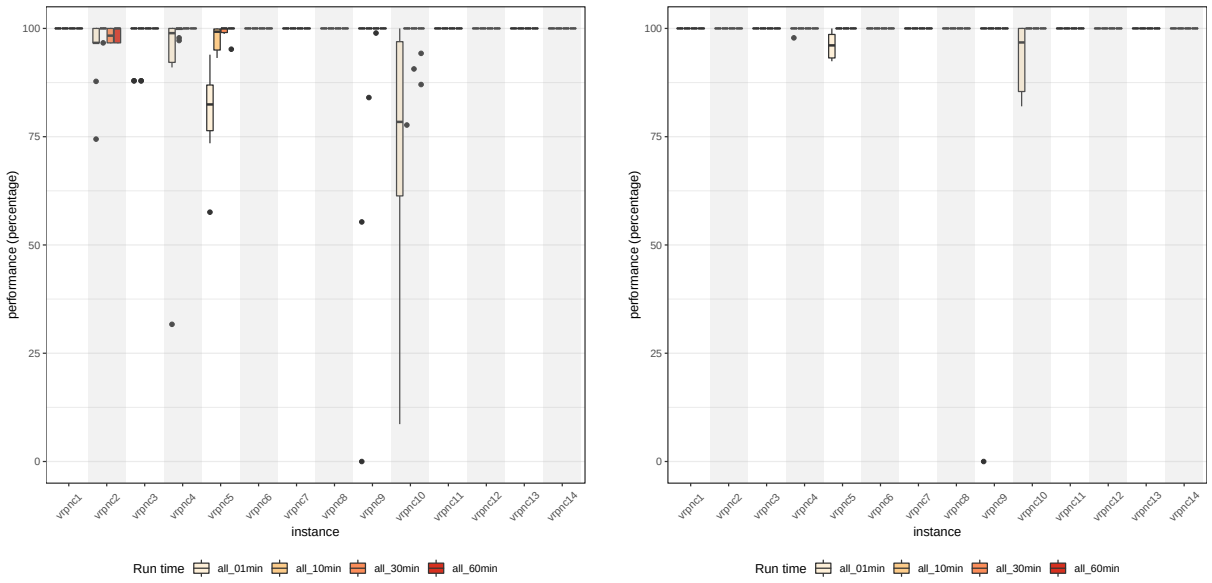
4.2. Computing time analysis

To further analyze the performance of the configuration *all*, we observe the results obtained after 1 minute, 10 minutes, 30 minutes and 1 hour, respectively. For that purpose, we consider the minimal number of vehicles ($m = m^\star$). The resulting solutions are represented on the four boxplots of Figure 10.



(a) % of the reference set found for each runtime

(b) % of the reference set within 1% distance from a found solution



(c) % of the reference set within 2% distance from a found solution

(d) % of the reference set within 3% distance from a found solution

Figure 10: Impact of runtime on the quality of the solution found: configuration *all*, $m = m\star$, runtime 1', 10', 30' and 60'.

For the easiest instances, a runtime of 10 or 30 minutes is satisfying. Indeed, Figure 10c shows that the reference set is within a 2% distance of the solutions found using these runtimes, except for instances vrpnc2, and vrpnc5 in which the number of missing solutions remains very low for all runs. Instance vrpnc14 which was hard for the *max* configuration is well solved even for low computation times when *leximax* operators are used.

It can be noted that, because dominated solutions are deleted at each insertion in the archive,

the number of archived solutions always remains low – always under 1000 in our experiments. Memory is not an issue in the studied instances of the *leximax*-VRP.

4.3. Price of fairness analysis

Figure 11 provides some managerial insight based on the Bertsimas et al. (2011) approach to valuate the price of fairness. To perform this analysis, we use the Pareto set approximation that is given by the reference sets computed previously for $m = m^\star$. For each instance, let x^* denote the cheapest solution found. Let also $z_1(x)$ be the cost of solution x, and $z'_2(x)$ be the length of the longest route in solution x ("max. duration"). The vertical axis on Figure 11 represents the *cost deterioration* of a solution s, given by the value $100 \times \frac{z_1(s)-z_1(x^*)}{z_1(x^*)}$. The horizontal axis represents the workload balancing improvement calculated as the relative decrease of the longest route: $100 \times \frac{z'_2(x^*)-z'_2(s)}{z'_2(x^*)}$. Figure 11 represents cost deterioration as a discrete function of max. duration improvement, as observed in the reference sets mentioned above.

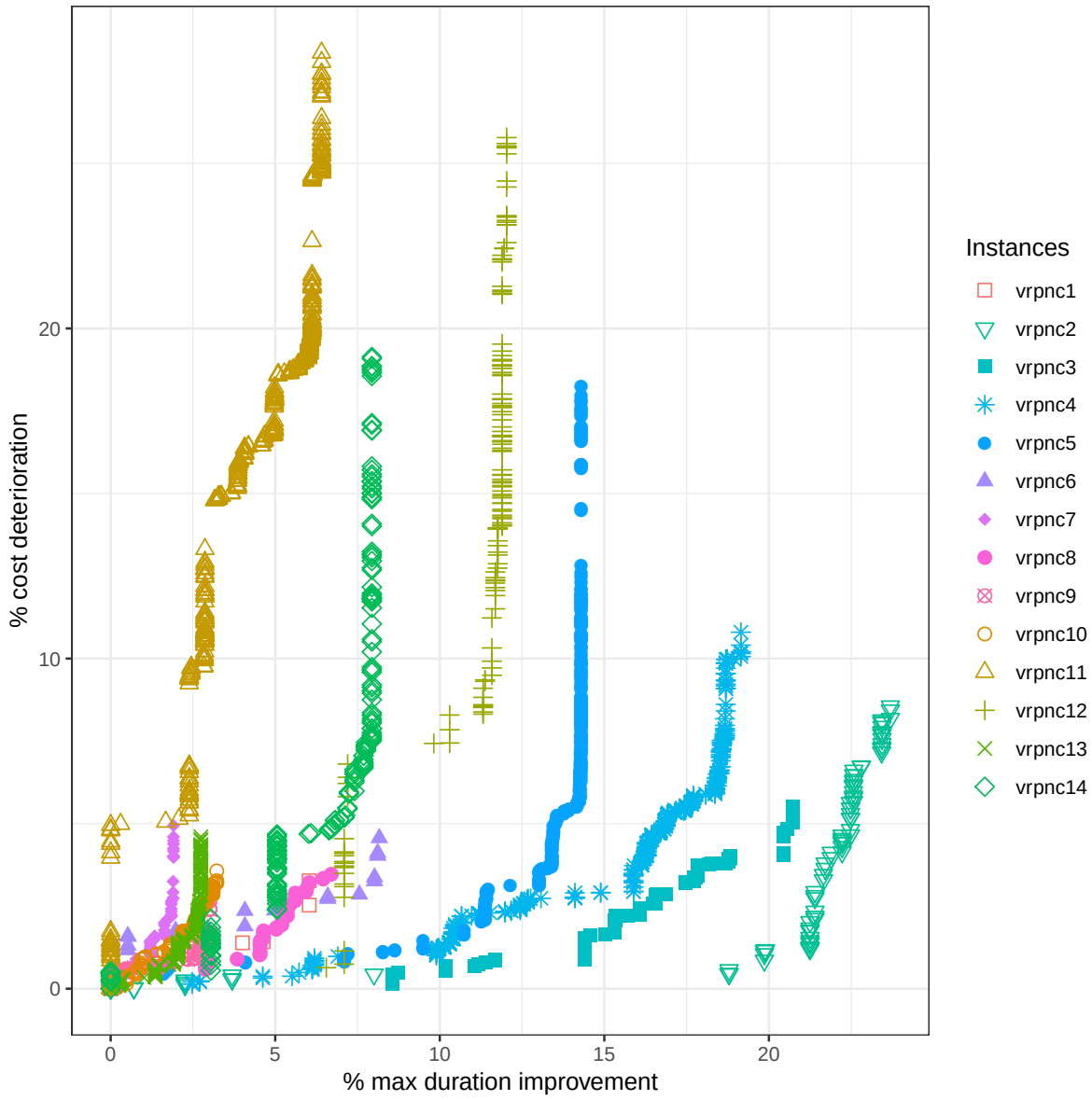


Figure 11: Normalized cost deterioration as a function of max. duration improvement for all instances.

It can be seen that the *price of fairness* clearly depends on the type of instance. On instances vrpnc2 to vrpnc5, which have no service duration and no maximum route duration specified, a good balancing improvement is obtained at the price of minor cost increase (for instance vrpnc2, the maximum route length is decreased by more than 20% if a 2% increase in cost is allowed).

If we observe instances vrpnc6 to vrpnc10, the potential for improvement remains weak due to the maximum route duration constraint. Still, we observe that a higher cost increase is necessary for a similar maximum route length improvement, in proportion. Nevertheless, this may simply come from the inclusion of service durations in route length which may bias the route balancing metric.

Considering instances vrpnc11, vrpnc12 and vrpnc14, finding improvements of the maximum

route duration seems considerably more costly than for other instances. These instances are clustered, so decreasing the duration of the longest route is likely to require having more routes go to multiple clusters, which in turns increases the cost of the solution more dramatically. Instance vrpnc13 is also clustered but has a tighter time limit than the others.

The maximum route duration metric only captures one aspect of workload balancing. Leximax cannot be represented graphically, but we perform a similar analysis using the *range* indicator instead of max. duration. We still only consider consistent solutions. This analysis is presented on Figure 12. The visualization offers a different perspective and seems less sensitive to service durations. Here, on most instances, a cost increase of up to 5 % allows for significant improvements in load balancing. Still, some sparsity is observed among instances and the results suggest that a good load balancing (w.r.t. range) may come at a high price for some instances.

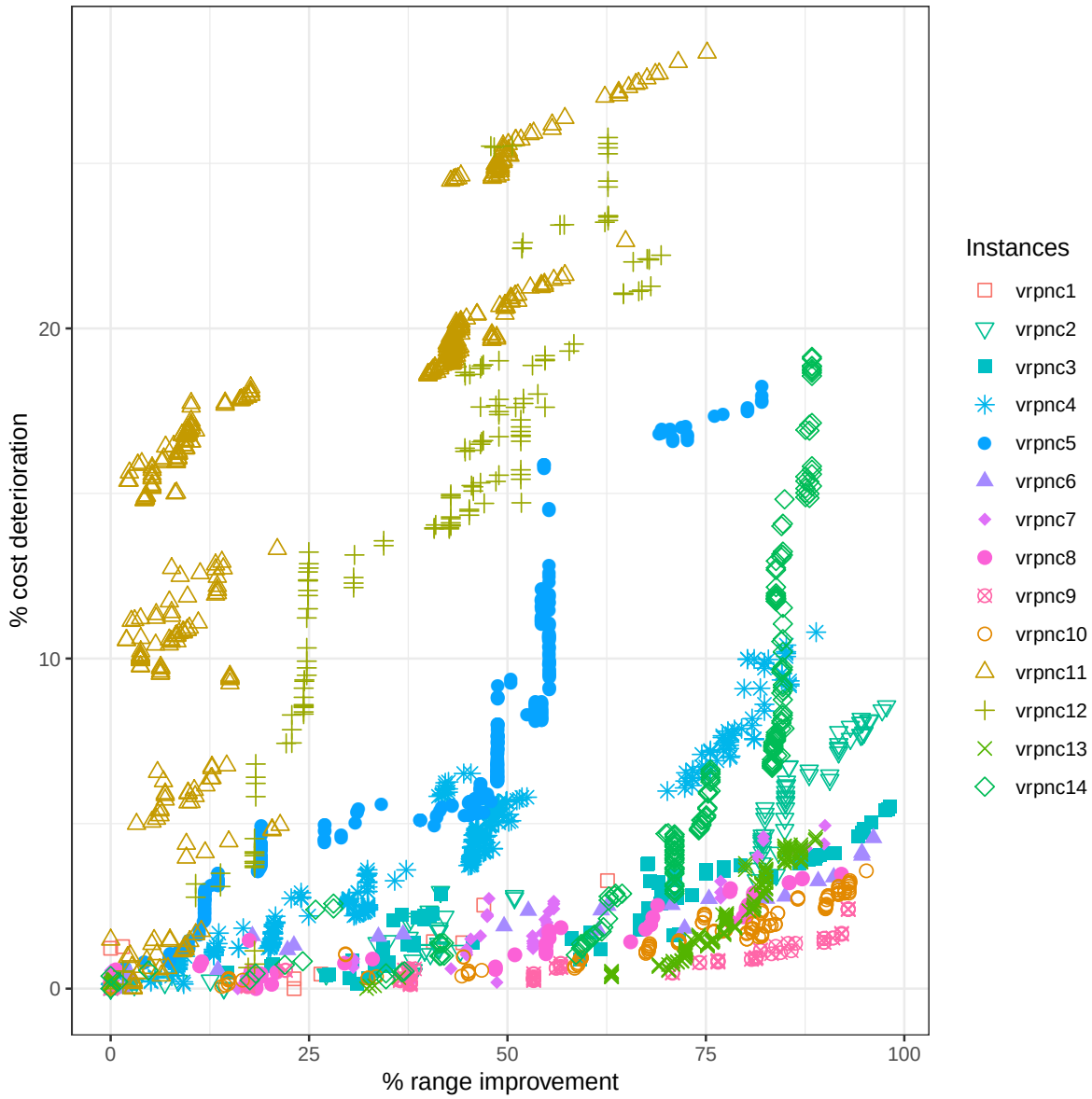


Figure 12: Normalized cost deterioration as a function of range improvement for all instances.

4.4. Equivalent solutions for the min-max objective function

A motivation for studying the *leximax*-VRP is that that the min-max approach fails to distinguish between solutions with the same longest route duration. We call such solutions min-max-equivalent. On Figure 13, we analyze the number of min-max-equivalent solutions in the reference set, by showing histograms of the distribution of this value for instances vrpnc1, vrpnc3, vrpnc5 and vrpnc11, ordered by increasing number of customers. Not surprisingly, the largest instances have the greatest number of min-max-equivalent solutions. Instance vrpnc5 has more than 150 distinct solutions with the same maximum route duration. It can also be observed that min-max-equivalent solutions are found not only between the most balanced solutions; in particular, instance vrpnc11 has more than 25 min-max-equivalent solutions for diverse values of maximum route duration.

Note that this analysis includes only non-dominated solutions in the solution sets found for the *leximax*-VRP.

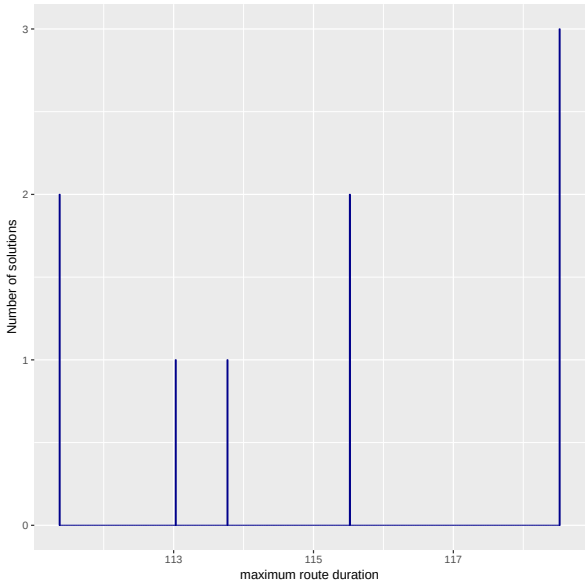
4.5. Solution analysis on an example

Let us illustrate the solutions of the *leximax*-VRP on the first instance of the Christofides et al. (1979) benchmark. For this instance, extremely stable results over all methods and runtimes suggest that the actual Pareto set may have been found.

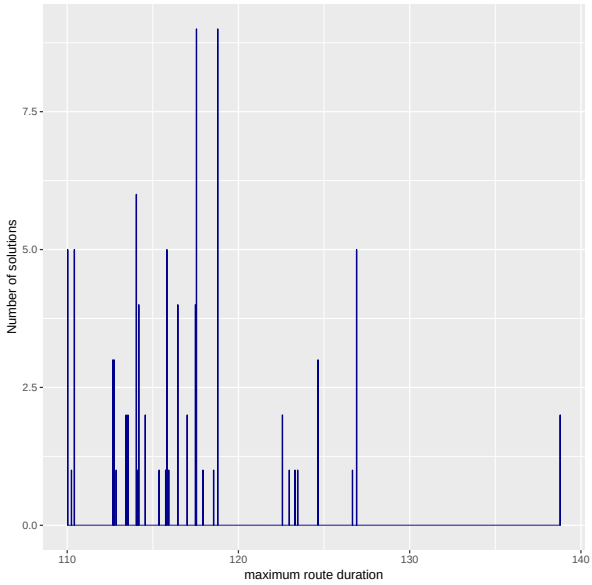
Table 5 presents the cost and the route durations for the nine solutions of the reference set. Route durations are sorted in a non–increasing order. These solutions are graphically represented on the space of the cost and maximum route duration dimensions on Figure 14.

Solution	Cost	Route Durations				
s_1	524.614	118.52	109.06	99.33	99.25	98.45
s_2	526.176	118.52	105.86	104.01	99.33	98.45
s_3	526.930	118.52	104.01	103.16	101.91	99.33
s_4	531.038	115.52	113.03	108.17	104.89	89.42
s_5	531.285	115.52	113.03	108.17	104.76	89.81
s_6	531.905	113.78	110.49	109.06	99.33	99.25
s_7	532.086	113.03	108.46	108.17	104.89	97.54
s_8	537.891	111.37	110.90	109.91	108.17	97.54
s_9	541.775	111.37	110.90	109.76	108.17	101.61

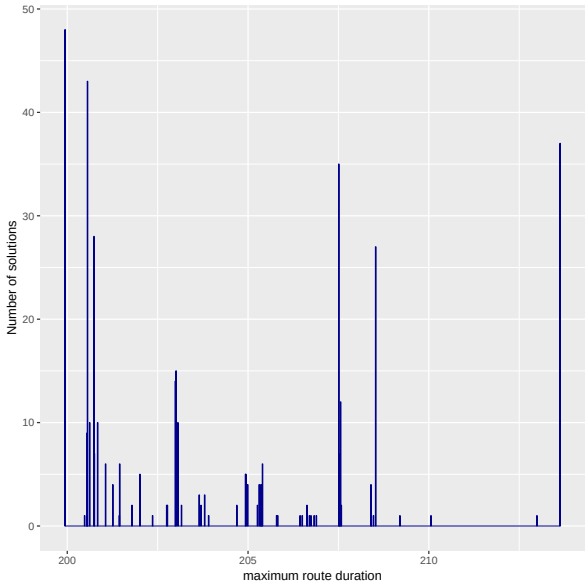
Table 5: Reference solutions for instance vrpnc1 of the Christofides et al. (1979) benchmark.



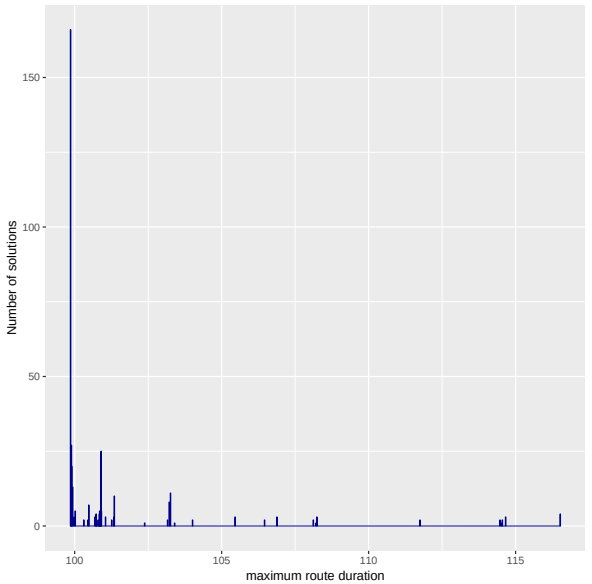
(a) Instance vrpnc1



(b) Instance vrpnc3



(c) Instance vrpnc11



(d) Instance vrpnc5

Figure 13: Distribution analysis for the maximum route duration of non dominated solutions in the *leximax*-VRP.

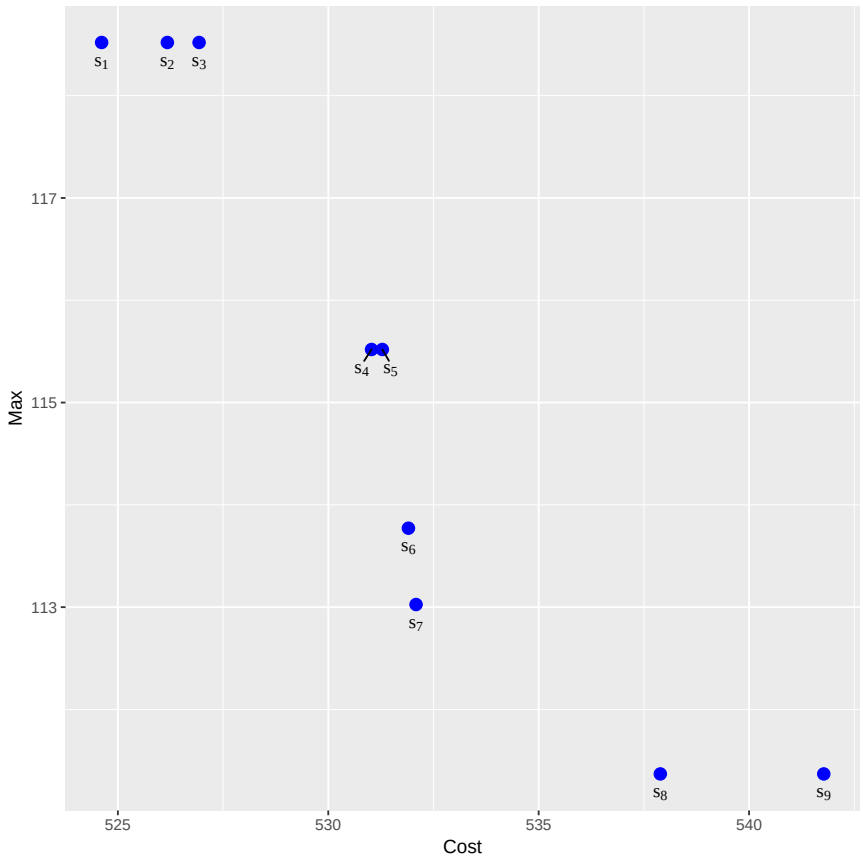


Figure 14: Graphical representation of the vrpnc1 reference set on the Cost and Maximum route duration dimensions (Max)

In this example, solutions s_1, s_2 and s_3 are equivalent with respect to the min-max objective function. So are s_4 and s_5 and s_8 and s_9. Still, these solutions offer different trade-off between cost and workload balance.

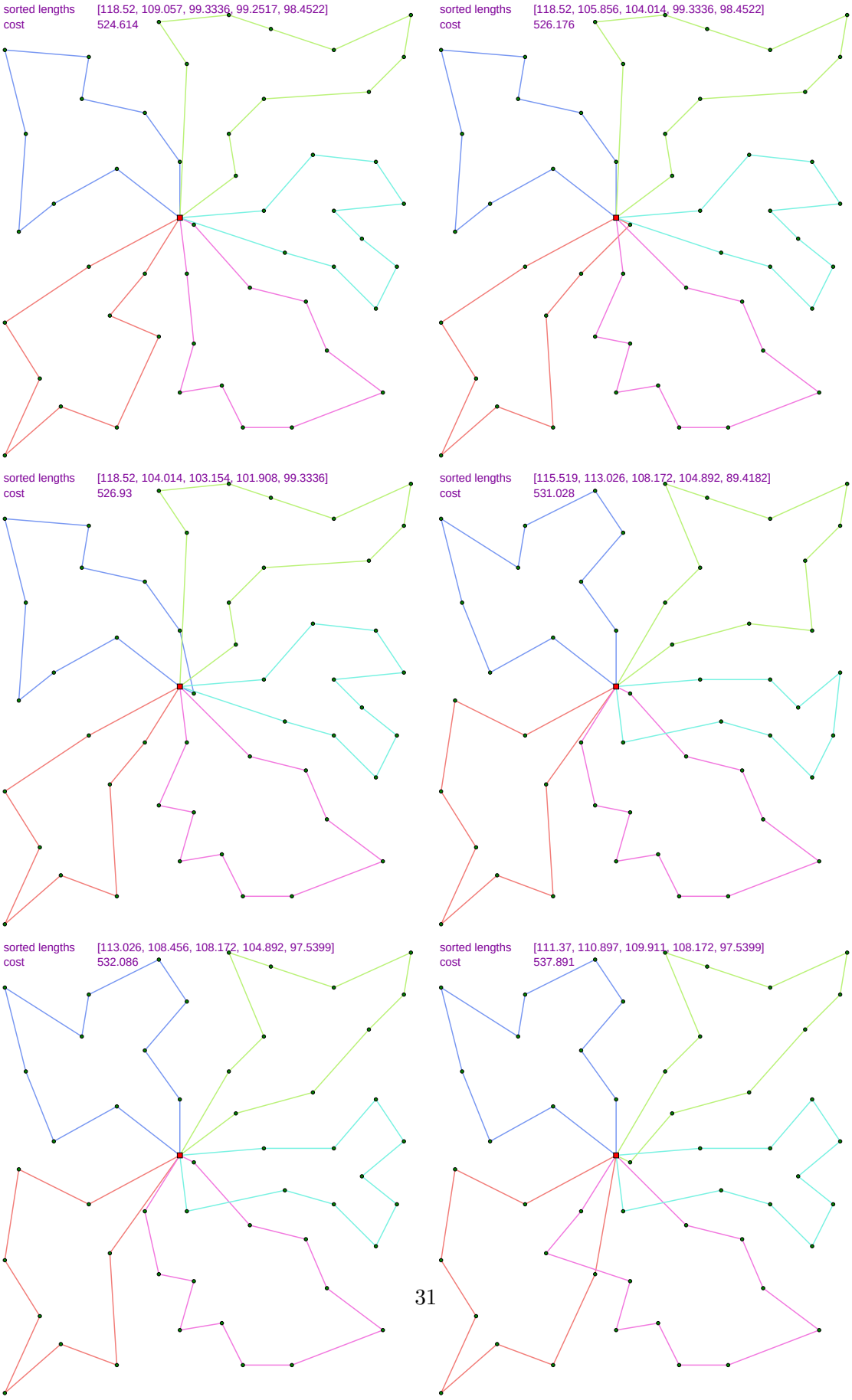


Figure 15: Six solutions of the vrpnc1 instance: s_1, s_2 (top row), , s_3, s_4 (middle row), s_7, s_8 (bottom row).

Figure 15 presents a graphical representation of the solutions s_1, s_2, s_3, s_4, s_7 and s_8 presented in Table 5. It can be seen that the slight variations between the three first solutions are obtained through different assignments of a customer that is close to the depot. Solutions s_4, s_7 and s_8 are obtained after more significant customer re-assignment. Between s_1 and s_8, the cost increases by 3.27%, while the duration of the longest route decreases by 6.03%. We note that the value of the *range* criterion changes non-monotonically, taking values 20.07, 20.07, 19.19, 26.10, 25.71, 14.53, 15.49, 13.83 and 9.76 when ordering the solutions by decreasing cost.

5. Conclusion

In this paper, we describe how the lexicographic minimax approach can be used to capture workload balancing concerns in vehicle routing, by considering the *leximax*-VRP. We tackle the *leximax*-VRP using multi-directional local search, thus approximating the Pareto set. In that context, we develop classical neighborhood operators for the cost objective, as well as a variety of operators for the workload balancing objective; in particular, we introduce new operators that explicitly consider the lexicographic minimax nature of the balancing objective. These operators are embedded in the multi-directional local search framework, and an experimental analysis shows that they offer a good performance: using them increases the quality of the produced solution sets. The practicality of operators explicitly considering the lexicographic minimax objective is thus demonstrated. Experiments also show that the more elaborate operators make a bigger difference when workload balancing is a predominant concern. For most instances, workload balancing can be significantly improved with a modest increase in cost. However, an analysis of our solution sets using previous criteria from the literature emphasizes that the criterion being used has an influence on the perceived price of fairness.

Integrating a lexicographic minimax approach with vehicle routing opens perspectives in logistics optimization. Above all, it shows that there exists an alternative to the min-max and range models, capturing workload balancing and fairness in a better way. While we provide a first explicit approach to tackle the *leximax*-VRP, we believe that there are rich perspectives in tackling this problem in different ways, especially in the design of exact methods for the *leximax*-VRP.

6. Aknowledgement

The authors would like to thank Peter Matl and Richard Hartl for fruitful discussions.

References

F. A. Behringer. A simplex based algorithm for the lexicographically extended linear maxmin problem. *European Journal of Operational Research*, 7(3):274–283, 1981.

L. Bertazzi, B. Golden, and X. Wang. Min–max vs. min–sum vehicle routing: A worst-case analysis. *European Journal of Operational Research*, 240(2):372–381, 2015.

D. Bertsimas, G. Lulli, and A. Odoni. The air traffic flow management problem: An integer optimization approach. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 34–46. Springer, 2008.

D. Bertsimas, V. F. Farias, and N. Trichakis. The price of fairness. *Operations Research*, 59(1):17–31, 2011.

I. Borgulya. An algorithm for the capacitated vehicle routing problem with route balancing. *Central European Journal of Operations Research*, 16(4):331–343, 2008.

S. Bouveret and M. Lemaître. Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364, 2009.

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester, 1979.

CVRPLIB, 2018. URL `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`.

D. Dubois, H. Fargier, and H. Prade. Fuzzy constraints in job-shop scheduling. *Journal of intelligent Manufacturing*, 6(4):215–234, 1995.

D. Dubois, H. Fargier, and H. Prade. Refinements of the maximin approach to decision-making in a fuzzy environment. *Fuzzy sets and systems*, 81(1):103–122, 1996.

M. Ehrgott. Discrete decision problems, multiple criteria optimization classes and lexicographic max-ordering. In *Trends in multicriteria decision making*, pages 31–44. Springer, 1998.

S. Eilon, C. D. Watson-Gandy, and N. Christofides. *Distribution management*. Griffin London, 1971.

B. L. Golden, G. Laporte, and É. D. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research*, 24(5):445–452, 1997.

E. E. Halvorsen-Weare and M. W. Savelsbergh. The bi-objective mixed capacitated general routing problem with different route balance criteria. *European Journal of Operational Research*, 251(2):451–465, 2016.

N. Jozefowiez, F. Semet, and E.-G. Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In *International Conference on Parallel Problem Solving from Nature*, pages 271–280. Springer Berlin Heidelberg, 2002.

N. Jozefowiez, F. Semet, and E.-G. Talbi. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195(3):761–769, 2009.

Ö. Karsu and A. Morton. Inequity averse optimization in operational research. *European Journal of Operational Research*, 245(2):343–359, 2015.

P. Lacomme, C. Prins, C. Prodhon, and L. Ren. A multi-start split based path relinking (MSSPR) approach for the vehicle routing problem with route balancing. *Engineering Applications of Artificial Intelligence*, 38:237–251, 2015.

T.-R. Lee and J.-H. Ueng. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, 29(10):646–657, 1999.

S. Liu and L. G. Papageorgiou. Multiobjective optimisation of production, distribution and capacity planning of global supply chains in the process industry. *Omega*, 41(2):369–382, 2013.

H. Luss. Equitable bandwidth allocation in content distribution networks. *Naval Research Logistics (NRL)*, 57(3): 266–278, 2010.

S. K. Mandal, D. Pacciarelli, A. Løkketangen, and G. Hasle. A memetic NSGA-II for the bi-objective mixed capacitated general routing problem. *Journal of Heuristics*, 21(3):359–390, 2015.

P. Matl, R. F. Hartl, and T. Vidal. Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science*, 52(2):239–260, 2017.

H. Moulin. *Fair division and collective welfare*. MIT press, 2004.

D. Nace and J. B. Orlin. Lexicographically minimum and maximum load linear programming problems. *Operations Research*, 55(1):182–187, 2007.

W. Ogryczak. On the lexicographic minimax approach to location problems. *European Journal of Operational Research*, 100(3):566–585, 1997.

W. Ogryczak and T. Śliwiński. On direct methods for lexicographic min-max optimization. In *International Conference on Computational Science and Its Applications*, pages 802–811. Springer, 2006.

W. Ogryczak, M. Pióro, and A. Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of telecommunications and information technology*, 3:43–56, 2005.

W. Ogryczak, H. Luss, M. Pióro, D. Nace, and A. Tomaszewski. Fair optimization and networks: A survey. *Journal of Applied Mathematics*, 2014.

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

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 99–127. Springer, 2019.

B. Radunović and J.-Y. L. Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking (TON)*, 15(5):1073–1083, 2007.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

S. Saliba. Heuristics for the lexicographic max-ordering vehicle routing problem. *Central European Journal of Operations Research*, 14(3):313–336, 2006.

A. K. Sen. *Collective choice and social welfare*. North Holland, 1970.

P. Shaw. Using constraint programming and local search methods to solve veh icle routing problems. In *Principles and Practice of Constraint Programming - CP98*, pages 417–431. Springer-Verlag, 1998.
P. Toth and D. Vigo. *The vehicle routing problem.* SIAM, 2002.
F. Tricoire. Multi-directional local search. *Computers & Operations Research*, 39(12):3089–3101, 2012.