An Evolutionary Algorithm for various extensions of the Arc Routing Problem: a novel splitting procedure and crossover operator

A Master Thesis for the Degree of
Econometrics and Management Science

by

P.A.J. Versfelt
375614

Supervisors:
Dr. R. Spliet (EUR)
T. van Dockum (Municipality of Rotterdam)
Co-reader:
R. Willemsen (EUR)

Erasmus School of Economics
ERASMUS UNIVERSITY ROTTERDAM

March 2022

1 Introduction

In the Netherlands, municipalities have many duties to fulfill on road networks, such as winter road maintenance, urban waste collection, street sweeping and road inspection. In the winter, roads may become dangerously slippery due to frost, ice or snow. To combat this, a de-icing agent (e.g., brine water) is spread on the roads by the municipality for safety reasons. The municipality of Rotterdam is the second largest municipality of the Netherlands with 990 km of roadways and 630 km of bike lanes. The municipality currently uses two distinct types of gritting routes to control slippery roads: curative gritting and preventive gritting.

Curative gritting is the process of removing snow and simultaneously gritting the snow-ploughed lane to remove any residual snow or ice. While preventive gritting is the process of gritting roads to prevent them from becoming slippery. When preventive gritting, snow ploughing is not applicable and a vehicle is able to grit multiple lanes simultaneously. The municipality of Rotterdam currently uses the same set of routes for preventive and curative gritting. Furthermore, a special set of routes is used for bridge gritting when bridges are susceptible to frost while roads are not. Bridges generally freeze earlier than roads, as the flowing water beneath bridges increases the humidity in the surrounding air and can result in frost due to condensation. The municipality of Rotterdam has many crucial infrastructure bridges, and thus has to employ preventive bridge gritting routes.

All gritting routes for the municipality of Rotterdam are manually drafted by in-house cartographers, dated from the early 2000s. The municipality is under pressure to manage her services efficiently, effectively and economically. Thus by creating separate routes for curative and preventive gritting, benefits can be achieved by gritting multiple lanes simultaneously. As the municipality personnel can resume with their remaining duties as soon as possible if the gritting process is finished earlier.

The municipality of Rotterdam is interested in a technique that can provide answers to the following problems:

1. The least number of vehicles needed to grit the selected bridges within a time limit;
2. The effect of servicing the bridges from a single, or multiple depots;
3. The trade-off between the duration of the longest route (makespan) and total duration of all routes.

This thesis is carried out in cooperation with the municipality of Rotterdam and focuses on the posed problems for the preventive gritting route optimisation by the municipality. The gritting routing problem can be modeled as capacitated arc routing problem (CARP) (Wøhlk, 2008), an NP-Hard combinatorial optimization problem (Golden and Wong, 1981). To this extent, first recent developments in the vehicle routing literature are implemented within a memetic algorithm (MA) for the CARP. Second, a novel splitting procedure for the heterogeneous fleet CARP is presented. Third, a novel crossover operator is presented based on the tabu search metaheuristic. Fourth, a MA for the multi-objective CARP (MO-CARP) is developed and a new clone management strategy is presented. Finally, in this thesis a variation of an existing extension of the CARP is presented: the Open-Ended CARP (OECARP). Within the OECARP, vehicles must depart from a depot, but need not end their routes at a depot.

The developed splitting procedure is compared to the label setting algorithm of Prins (2009). On average, the proposed splitting procedure produced speed-ups over the label setting algorithm. Where larger speed-ups were observed on large instances and on fleets consisting of multiple vehicle types. Hereafter, a crossover analysis was performed for multiple classical crossover operators. The crossover analysis was performed on CARP instances, where the analysis showed that the novel crossover operator outperformed other crossovers in both solution value and running time. Subsequently, the novel crossover operator is incorporated within the developed MA. The MA improved 6 out of 34 best known solutions (BKS) of the largest instance sets of the CARP and outperformed the current state-of-the-art algorithm (Vidal et al., 2014) in both solution value and running time. Hereafter, the results for the OECARP and its extensions are presented. Finally, the MO-CARP algorithm results are provided. The developed MO-CARP algorithm outperforms the current state-of-the-art MO-CARP algorithm DMEANS-2 (Zhang et al., 2020), as improvements on the Pareto-Fronts of multiple instances were identified. Moreover, on 18 of the 24 instances better performance measure values were obtained for the hypervolume and inverted generational distance. Furthermore, the proposed clone management strategy performed well for smaller instances, but did not outperform existing clone management strategies on the largest instances.

This thesis is structured as follows. In Section 2 the researched problem is described in more detail. Hereafter, the relevant academic literature is discussed in Section 3. Next, a description of the methodology is presented in Section 4. In section 5, computational experiments are provided. Finally, Section 6 contains the conclusion and discussion.

2 Problem description

The CARP can be described as a connected graph $G = (V, A)$ in which V and A denote the set of nodes and the set of (directed) arcs respectively. A subset $R \subseteq A$ of arcs require service by a vehicle. Edges are modeled as two directed arcs within the graph G. Let $u' = (j, i)$ be the reverse arc of $u = (i, j)$ for an edge (i, j), where $i, j \in V$. Thus if u (u') is serviced, then u' (u) does not require service if u' is the reverse of u. All arcs $u \in A$, where $u = (i, j)$ and $i, j \in V$, can be traversed any number of times. Furthermore, each arc $u \in A$ has a traversal cost $c(u) > 0$ and demand $q(u) \geq 0$. Moreover, let $tl(u) = i$ and $hd(u) = j$ denote the tail and head node respectively of arc $u = (i, j)$. Then $\text{dist}(hd(u), tl(v))$ denotes the shortest distance to traverse from the head node of arc u to the tail node of v, where $u, v \in A$. A fleet of K identical vehicles of capacity W are based at a depot node $\sigma \in V$. The CARP is tasked with determining a set of routes that minimize the total cost, satisfying the following constraints:

1. Each vehicle starts and ends its route at the depot σ;
2. Each required arc $r \in R$ is serviced exactly once;
3. The total demand handled by any vehicle (i.e., load) does not exceed W.

In this work, a routing problem for preventive gritting is addressed for a Dutch municipality. The municipality is bounded by law, that all required streets must be serviced within a time limit. Thus, the deadheading time occurred by a vehicle when returning to its depot after servicing its last arc can be ignored when determining the least number of vehicles necessary. Thus, the constraint requiring that each vehicle must start and end its route at the depot σ is relaxed. Vehicles are only required to start its route at the depot, creating the Open-End CARP (OECARP). Hereafter, the problem is extended by allowing vehicles within the fleet to be stationed at multiple-depots, resulting in the Multi-Depot OECARP (MDOECARP). To this extent, the unique depot is replaced by a set of depots $D \subseteq V$. Finally, the fleet of the municipality consists of multiple vehicle types, hence the problems can be extended by a heterogeneous fleet of various vehicle types $T = \{t_1, t_2, ..., t_{|T|}\}$. While vehicles can differ in multiple dimensions, this research only focuses on differences in capacities. Therefore, let W_t denote the capacity and a_t denote the number of available vehicles of type $t \in T$.

These extensions in unison create the Heterogeneous Fleet Multi-Depot Open-End CARP (HFMDOECARP). The HFMDOECARP is concerned with a single objective: minimizing the total cost. However, decision makers are also interested in completing the gritting as soon as possible (otherwise known as the makespan) while also minimizing the total cost, as the gritting vehicles of the municipality of Rotterdam are operated by employees that normally perform other tasks (e.g., waste collection). The addition of multiple objectives to the CARP results in the Multi-Objective CARP (MO-CARP).

3 Literature review

In 1735 Euler posed the Königsberg Bridges problem, the first arc routing problem. Since its introduction, arc routing problems arise in many operations on street networks (Golden et al., 2008). For instance, mail delivery, street sweeper, waste collection, gas pipeline inspection and electric power line inspection (Xing et al., 2010). The counterpart of the Arc Routing Problem (ARP) is the Vehicle Routing Problem (VRP). In the ARP, arcs are in need of service, while in the VRP only nodes are considered. Arc routing and node routing problems seem similar at first sight, but are distinctly different. Various traversal orientations of service (i.e., reverse arcs) within the graph have to be considered in the ARP, while these orientations of service are not present in the VRP (Vidal, 2017).

While transformations exist between arc routing and vehicle routing problems by transferring the demand to additional nodes, these operations increase the instance size substantially. Longo et al. (2006) show that an original problem instance for the CARP of n required edges, can be transformed into a vehicle routing problem with $2n + 1$ nodes. Thus for larger instances, it is favourable to solve the arc routing problem due to the increased instance size.

The Capacitated ARP (CARP) was formally introduced in 1981 and is proven to be an NP-Hard problem (Golden and Wong, 1981). Since its introduction, exact methods have been proposed for the CARP. Several formulations exist within the CARP literature, the first was proposed by Golden and Wong (1981) and used an exponential number of variables and constraints. Hereafter, Belenguer (1990) proposed a two-index formulation. This formulation has been applied by using the Branch-and-cut and the branch-and-bound (Welz, 1994) approach. A single-index formulation in combination with a cutting-plane algorithm was published by Belenguer and Benavent (2003). Finally, a set-covering formulation combined with a cut-and-column generation technique was proposed by Gómez-Cabrero et al. (2005).

In their review of exact methods for solving the CARP, Belenguer et al. (2015) show that the recent most successful exact solution approaches are cutting-plane methods in combination with the two-index formulation or cut-and-column-generation methods and the set-covering formulation. These methods provide optimal solutions for most smaller benchmark instances (less than 150 required arcs). However, depending on the extension of the CARP, exact methods become harder to solve (Belenguer et al., 2015). Therefore, for real-world problems with multiple extensions to the CARP, heuristic methods are currently the preferred method (Prins et al., 2014).

Initially, problem specific heuristics were employed to solve the CARP. The most commonly used heuristics for the CARP are the path-scanning algorithm (Golden et al., 1983), Augment-merge (Golden and Wong, 1981), Augment-insert (Pearn, 1991) and Ulusoy's tour splitting algorithm (Ulusoy, 1985). Recently new problem specific heuristics have been proposed, for instance the Double Outer Scan Heuristic (Wøhlk, 2005), the ellipse-rule path-scanning algorithm (Santos et al., 2009) and the path-scanning with efficiency rule (Arakaki and Usberti, 2019). These heuristics, in spite of their simplicity, construct good routes in practice. Nevertheless, better solutions can be obtained by using metaheuristics.

A simulated annealing approach was used by Eglese (1994) to solve the CARP specified to a winter gritting problem. Furthermore, various Tabu Search (TS) algorithms have been proposed to solve the CARP (Eglese and Li, 1996), (Hertz et al., 1999) and (Corberán et al., 2000). A Variable Neighborhood Descent (VND) algorithm (Hertz and Mittaz, 2001), Guided Local Search (GLS) (Beullens et al., 2003) and Ant Colony Optimization (ACO) technique (Lacomme et al., 2004b) have as well been applied to the CARP. However, since the first Genetic Algorithm (GA) was presented by Lacomme et al. (2001) and the Memetic Algorithm (MA) of Lacomme et al. (2004a), population based algorithms have been among the best performing algorithms for the CARP (Golden et al., 2008).

The GA and MA of Lacomme are based on the route-first-cluster-second approach. This procedure was first proposed by Beasley (1983), in which the vehicle capacity is temporary relaxed and a giant tour is constructed by solving the traveling salesman problem. Hereafter, the giant tour is split into minimum cost routes satisfying the capacity constraints and hence constructing a CARP solution. The counterpart of the route-first-cluster-second is the cluster-first-route-second approach (Gillett and Miller, 1974). Within this approach, customers are first clustered together and hereafter the CARP is solved for each individual cluster. Recently, the route-first-cluster-second has been given more attention, due to its advantageous flexibility, smaller solution space and efficiency (Prins et al., 2014).

In the route-first-cluster-second MA of Lacomme et al. (2004a), a chromosome is defined as a sequence of required arcs with implicit shortest paths between arcs. Chromosomes are created by random generation, crossover or by improving an existing solution. A random chromosome is created by classical CARP constructive heuristics or by randomly ordering the required arcs and subsequently calling a procedure that splits the giant tour into routes to produce a solution. Offspring chromosomes are created by utilizing a crossover operator in combination with binary tournament selection. After the crossover, offspring chromosomes are evaluated using the split procedure. The split procedure within the MA is tasked with splitting the giant tour into routes to construct the least cost solution of a chromosome. To this extent, an auxiliary graph in combination with the Bellman-Ford algorithm is utilized. The split procedure applied in their study has a time-complexity of $\mathcal{O}(n^2m)$ for n required arcs and a fixed homogeneous fleet of size m. Finally, a fraction of the created offspring undergoes a local search procedure in an effort to improve the solution quality of the solution.

Recently, multiple improvements have been made to the MA of routing problems. First, the splitting procedure for VRPs was improved by Vidal (2015) by exploiting the Monge property present on the auxiliary graph. Thereby reducing the time complexity from $\mathcal{O}(n^2m)$ to $\mathcal{O}(nm)$. Second, advancements in population and diversity management methods for vehicle routing have been proposed by Vidal et al. (2014) in their Unified Hybrid Genetic Search (UHGS) method. Within their method, two subpopulations are managed: a feasible and infeasible population. By allowing infeasible solutions, a larger solution space can be explored in an effort to obtain better solutions. Chromosomes within the UHGS are evaluated by their total cost and a penalized cost if the capacity constraints are violated. Furthermore, chromosomes are compared relatively by their feasibility and contribution to population diversity. Within the UHGS, a fraction of infeasible CARP solutions are repaired and hereafter added to the correct population regarding the feasibility. Moreover, when a number of iterations have passed without improvement, a diversification phase is called. In this phase, a proportion of the population is retained, while others are replaced by new chromosomes. To date, their algorithm produced the best results on the classical benchmark CARP instances.

A wide variety of extensions have been proposed to the CARP in recent years. The CARP with intermediate facilities (CARPIF) extents the CARP by introducing dump sites. The CARPIF have been tackled by multiple metheuristics such as the ACO (Ghiani et al., 2001) and VND (Polacek et al., 2008). The CARP with Time Windows (CARPTW) is tasked with servicing arcs within a certain time period. The CARPTW was tackled using the Greedy Randomized Adaptive Search Procedure (GRASP) (Luiz Usberti et al., 2013), a heuristic based on the Path-Scanning algorithm (Wøhlk, 2005) and a GA by Ramdane-Cherif (2006). When a long time period is considered by the CARP such that required arcs must be serviced more than once, the problem can be described as a Periodic CARP (PCARP). A GA is employed by Lacomme et al. (2005) and a Scatter Search method by Chu et al. (2006) are applied for solving the PCARP. The Stochastic CARP (SCARP) differs from the original CARP due to its uncertainty of the demand of the required arcs. Fleury et al. (2004) propose a MA to solve the SCARP. The Multi-Depot CARP (MCARP), where multiple depots are available for the fleet of vehicles, has been solved using GAs (Vidal, 2017) and (Xing et al., 2010), the ACO by Kansou and Yassine (2009) and TS (Amberg et al., 2000). Recently the CARP was extended by relaxing the constraint that vehicles need to start and end their route at a depot, the Open CARP (OCARP). The OCARP was introduced and solved using a MA by Liu et al. (2010). Usberti et al. (2011) solved the same OCARP using a Path-Scanning based heuristic. While the OCARP seems similar to the OECARP, they substantially differ as in the OECARP restricts routes to start at a depot. The OECARP studied in this thesis, has not yet been tackled by previous academic studies. Finally, the CARP can be extended by an Heterogeneous Fleet (HFCARP). The HFCARP has been addressed by a MA (Prins, 2009), GLS (Liu et al., 2014), VND (Pia and Filippi, 2006) and a cluster-first-route-second approach by Ghiani et al. (2005).

The CARP is only concerned with a single objective: the total cost. While the Multi-Objective CARP (MO-CARP) optimizes multiple objectives such as makespan and the total cost. The MO-CARP therefore creates a Pareto-front which represents CARP solutions that are not dominated by any other solutions. In the Pareto-front, no solution can improve any objective value without degrading another objective value. Lacomme et al. (2006) was the first study to be published on the MO-CARP. In their study, they constructed a Pareto-front for the makespan and total cost by adopting the general framework of the Non-dominated Sorting Genetic Algorithm (NSGA-II) specified to the CARP. Hereafter, Mei et al. (2011) published their decomposition-based MA. Within their MA, the population is decomposed into subpopulations that evolve separately. Their MA was later improved by Zhang et al. (2020), by incorporating the NSGA-II to ensure diversity within each subpopulation. An Iterated Local Search procedure was proposed by Grandinetti et al. (2012) for solving the MO-CARP. All these studies, studied the MO-CARP without imposing the fleet size m, as the two objectives are conflicting (Lacomme et al., 2006). However, in a real-world application of the MO-CARP, an unlimited fleet is unrealistic. Therefore, in this thesis, the fleet size is not relaxed for the MO-CARP.

4 Methodology

In this section, the MA framework for the CARP is introduced and a new crossover operator is proposed for solving the CARP. Hereafter, extensions to the CARP are discussed and a new splitting procedure for heterogeneous vehicles is proposed. Finally, a multiple population NSGA-II framework is presented for solving the multi-objective CARP and its extensions.

4.1 Memetic Algorithm for the CARP

The main features of the MA are its chromosome structure, population management, evaluation of chromosomes, crossover, mutation and education method.

Chromosome structure. In the proposed MA, a chromosome $\mathcal{S} = \{a_1, a_2, ..., a_n\}$ in the population $\mathcal{P}$ is denoted as a sequence of n required arcs, where $a_i \vee a_i' \in R \; \forall i \in \{1, ..., n\}$, without explicit depot visits (Lacomme et al., 2004a). Each chromosome $\mathcal{S}$ therefore represents a giant tour that visits each required arc with implicit shortest distances between consecutive arcs. These shortest distances between all arcs in the graph G can be precalculated using Dijkstra's algorithm in combination with a Fibonacci heap.

While $\mathcal{S}$ is not equal to a CARP solution, $\mathcal{S}$ can be converted into a solution $\mathcal{T}(\mathcal{S}) = \{\mathcal{T}^1, ..., \mathcal{T}^K\}$, where K denotes the number of available vehicles. Each route $\mathcal{T} = \{\sigma, \mathcal{T}_1, ..., \mathcal{T}_{|\mathcal{T}|}, \sigma\} \in \mathcal{T}(\mathcal{S})$ represents a route visiting $|\mathcal{T}|$ arcs, where $\mathcal{T}_j \in R$ and the depot is denoted by $\sigma \in V$. Each route $\mathcal{T}$ has a corresponding cost and load defined by Equation 1 and Equation 2 respectively. Furthermore, a solution $\mathcal{T}(\mathcal{S})$ can be converted back into a chromosome $\mathcal{S}$ by creating a concatenated list of subsequent arcs of all routes.

$$\text{cost}(\mathcal{T}) = \text{dist}(\sigma, tl(\mathcal{T}_1)) + \sum_{j=1}^{|\mathcal{T}|-1} [c(\mathcal{T}_j) + \text{dist}(hd(\mathcal{T}_j), tl(\mathcal{T}_{j+1}))] + \text{dist}(hd(\mathcal{T}_{|\mathcal{T}_i|}), \sigma) \tag{1}$$

$$\text{load}(\mathcal{T}) = \sum_{j=1}^{|\mathcal{T}|} q(\mathcal{T}_j) \tag{2}$$

$$\phi^{\text{COST}}(\mathcal{T}(\mathcal{S})) = \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{S})} \text{cost}(\mathcal{T}) + \delta^W \max[0, (\text{load}(\mathcal{T}) - W)] \tag{3}$$

Following the framework of the MA of Vidal et al. (2014), solutions may be infeasible. The infeasible solution space provides the possibility that feasible solutions can be constructed from previously infeasible solutions. Hence, a larger feasible solution space can be explored by including infeasible solutions. Therefore, each chromosome $\mathcal{S}$ has a corresponding penalized cost $\phi^{\text{COST}}(\mathcal{T}(\mathcal{S}))$, consisting of a traversal cost and penalized load cost defined by Equation 3. The penalized cost is equal to the excess load, where W denotes the capacity of the vehicle, multiplied by a penalty factor δ^W.

Population management The general framework of the proposed MA is depicted in Algorithm 1, a framework based on the study of Vidal (2017) appended with a decomposition phase based on the study of Santini et al. (2021). Within this framework, two subpopulations are maintained, a feasible and infeasible subpopulation. In the initialization, chromosomes are constructed by ordering the required arcs in random order. Hereafter the chromosomes are evaluated and each chromosome is included into the appropriate subpopulation based on its feasibility. Furthermore, each chromosome in the subpopulation keeps track of its *broken pair distance* $d(\mathcal{S}, \mathcal{S}_2)$ to all other chromosomes in the subpopulation. Where $d(\mathcal{S}, \mathcal{S}_2)$ denotes the number of adjacent arcs in $\mathcal{S}$ that are no longer adjacent

Algorithm 1: Memetic algorithm

```
1  Initialize subpopulation $\mathcal{P}_f$ and $\mathcal{P}_{inf}$
2  while number of iterations without improvement $\leq it_{max}$ and time $\leq T_{max}$ do
3      Apply binary tournament to obtain parents $P_1$ and $P_2$
4      Create offspring O from $P_1$ and $P_2$ by crossover
5      Evaluate O by splitting procedure
6      Improve O by local search procedure
7      Adjust penalty parameter
8      if O is infeasible then
9          Insert O into $\mathcal{P}_{inf}$
10         Repair O with probability $P_{REP}$
11     else
12         Insert O into $\mathcal{P}_f$
13     if $|\mathcal{P}_i| = \mu + \lambda$, where $i = f, inf$ then
14         Prune subpopulation $\mathcal{P}_i$ to size $\mu$
15     if best solution has not improved in $it_{div}$ iterations then
16         Prune both subpopulations to size $\frac{\mu}{3}$ (diversification)
17         Initialize $\mu$ new chromosomes
18     if number of non-improving iterations $= k \cdot it_{dec}$ where $k \in \mathbb{N}$ then
19         Select chromosome D among best 25% feasible solutions
20         Decompose D into subproblems
21         Solve each subproblem by applying Algorithm 1
22         Recreate chromosome and insert into correct subpopulation
23 Return best feasible solution
```

in $\mathcal{S}_2$ (Prins, 2009). For instance, $\mathcal{S} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $\mathcal{S}_2 = \{a_4, a_5, a_6, a_2, a_3, a_1\}$ then $d(\mathcal{S}, \mathcal{S}_2) = 2$ as (a_1, a_2) and (a_3, a_4) are broken. Whenever a chromosome enters or departs from a subpopulation, all broken pairs distances lists of chromosomes are updated. The procedure to update the broken pairs distances list for all chromosomes, whenever a new chromosome is added to a subpopulation, is executed in $\mathcal{O}(n(\mu + \lambda))$ time.

The algorithm produces new chromosomes each iteration and terminates whenever the maximum time has been reached or if no improvement was found in a defined number of iterations. In each iteration, a binary tournament is applied on the union of both subpopulations $\mathcal{P}_f \cup \mathcal{P}_{inf}$. In the binary tournament, two chromosome candidates are relatively compared using their fitness and the best candidate (i.e., lowest fitness) is selected as parent. The fitness function consists of two elements: the diversity contribution rank and a cost rank. The average diversity contribution $dc(\mathcal{S})$ of chromosome $\mathcal{S} \in \mathcal{P}$ is defined by Equation 4, where $\mathcal{N}(\mathcal{S})$ denotes the set of n_{CLOSEST} chromosomes in terms of their broken pairs distance. Then the fitness of an individual $\mathcal{S}$ is equal to $f(\mathcal{S})$, see Equation 5. Where $r_c(\mathcal{S})$ denotes the rank of $\phi^{\text{COST}}(\mathcal{T})$ and $r_{dc}(\mathcal{S})$ the rank $dc(\mathcal{S})$ of chromosome $\mathcal{S}$ within subpopulation $\mathcal{P}$. The fraction $(1 - \frac{n_{\text{ELITE}}}{|\mathcal{P}|})$ ensures that the best $n_{\text{ELITE}} \geq 1$ are not removed from the subpopulation, even though their diversity might be poor (Vidal et al., 2014).

$$dc(\mathcal{S}) = \frac{\sum_{S_2 \in \mathcal{N}(\mathcal{S})} d(\mathcal{S}, \mathcal{S}_2)}{n_{\text{CLOSEST}}} \tag{4}$$

$$f(\mathcal{S}) = r_c(\mathcal{S}) + (1 - \frac{n_{\text{ELITE}}}{|\mathcal{P}|}) r_{dc}(\mathcal{S}) \tag{5}$$

After the selection of parents, a crossover operator is applied to create an offspring. Hereafter, the offspring O is evaluated to produce a set of routes $\mathcal{T}(O)$. Whenever $\mathcal{T}(\mathcal{O})$ is infeasible (i.e., the capacity constraint is not adhered to), the solution is repaired with a probability p_{REP} and hereafter added to the correct subpopulation. To repair a solution, the penalty factor δ^W is temporarily multiplied by 10. The resulting solution is subsequently placed in the appropriate subpopulation. Furthermore, the capacity violation parameter, $\delta^W \geq 0$, can be adapted to obtain approximately a ratio of $\theta \in [0, 1]$ feasible solutions. The parameter is increased (decreased), whenever the ratio drops too far below (above) a user desired ratio ξ^W.

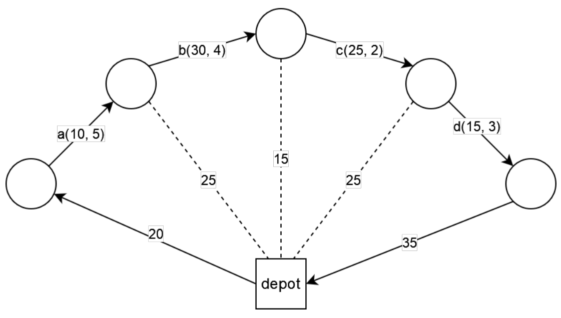
Each subpopulation is managed such that at least μ and at most $\mu + \lambda$ chromosomes are included in each subpopulation. To this extent, whenever a new chromosome is created and its addition to a subpopulation would violate the maximum size of that subpopulation, the subpopulation is pruned down to μ chromosomes. First, *clones* are removed. A clone is a chromosome $\mathcal{S}$ that has a *broken pairs distance* equal to 0 (i.e. $\exists\, \mathcal{S}_2 \in \mathcal{P}$ such that $d(\mathcal{S}, \mathcal{S}_2) = 0$). Clones are iteratively removed, where the worst clone in terms of its fitness is first removed. Hereafter, the chromosomes with the worst fitness values are iteratively removed, until μ chromosomes remain in the subpopulation. Furthermore, whenever no improvement was determined in it_{div} iterations, a survival phase is triggered. In the survival phase, the worst chromosomes in terms of fitness are removed until $\frac{\mu}{3}$ chromosomes remain in the subpopulation. This process ensures that the MA is able to escape local minima by switching between diversification and intensification of the solution space. Finally, a decomposition phase is initiated whenever the algorithm could not identify any improvements in it_{dec} iterations.

Chromosome evaluation. After the crossover operator, a chromosome represents a sequence of n required arcs (i.e., a giant tour). Thus, to obtain a set of routes, a splitting procedure must be used. The aim of the splitting procedure is to create a set of least cost routes $\mathcal{T}(\mathcal{S})$ that contain consecutive visits from the giant tour. This procedure can be formulated as constructing the shortest path between node 0 and n in a directed cyclic graph (DAG) $H = (V', A')$, where $V' = (0, ..., n)$ and $(i, j) \in A'$ for $i < j$ and $i, j \in V$. Each node $i \in V'$ is mapped to a required arc in $\mathcal{S} = \{a_1, a_2, ..., a_n\}$. Let $a(i) = u$ denote the mapped node of $i \in V'$ to arc $u \in \mathcal{S}$. Where node 1 is mapped to the first required arc and node 2 is mapped to the second required arc of the chromosome and so on (i.e., $a(i) = a_i \; \forall i = \{1, ..., n\}$). Therefore, each arc $(i, j) \in A'$ corresponds to a route starting from the depot, visiting the mapped required arcs corresponding to nodes $i + 1$ to j and returning to the depot. Then let $c(i, j)$ denote the cost of an arc $(i, j) \in A'$. The shortest path of node 0 to n on H can be solved in $\mathcal{O}(n^2 m)$ using the Bellman-Ford algorithms for n required arcs and m vehicles (Lacomme et al., 2004a). Figure 1 depicts an example of the splitting procedure for an arbitrary chromosome containing 4 arcs.

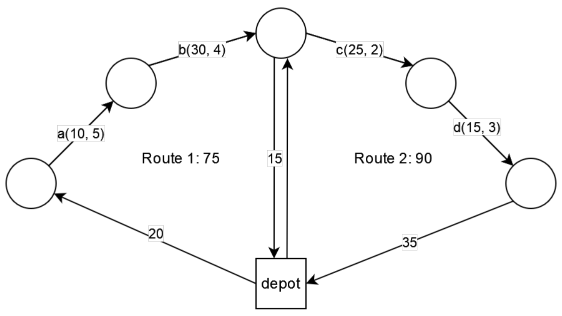
Recently, Vidal (2015) noted that the splitting procedure could be solved in $\mathcal{O}(nm)$ by exploiting the following property present in the auxiliary graph H to eliminate predecessors of labels generated in their labeling algorithm:

$$\begin{gathered} \text{for all } 0 \leq i_1 < i_2 < n, \text{ there exists } K \in \mathbb{R} \text{ such that} \\ c(i_1, j) - c(i_2, j) = K \text{ for all } j > i_2 \text{ such that } (i_1, j) \in A'. \end{gathered} \tag{6}$$

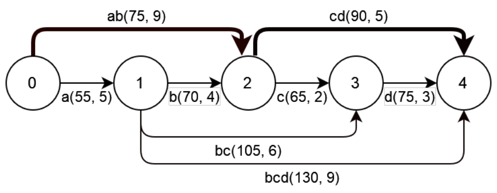
The labeling algorithm of Vidal (2015) for VRP utilizes auxiliary data structures to ensure constant time cost lookup, preprocessed in $\mathcal{O}(n)$ time at the start of the algorithm. To this extent, $D[i]$ and $Q[i]$ define the cumulative distance between arcs and the cumulative capacity of the mapped arcs to the nodes up to $i \in \{1, ..., n\}$ respectively. Let $d_{i,j} = dist(hd(u), tl(v))$ denote the cost of traversing from the end of the arc $u = a(i)$ for $i \in V'$



(a) Giant tour $\mathcal{S}$ = {a, b, c, d}, costs and capacities between brackets



(c) Optimal split, cost = 185



(b) Auxiliary graph H with maximum capacity W = 10 (shortest path in bold)

Figure 1: Example of splitting procedure.

to the start of arc $v = a(j)$ for $j \in V'$. Furthermore, let $d_{\sigma,j} = dist(\sigma, tl(v))$ and $d_{i,\sigma} = dist(hd(u), \sigma)$ denotes the distance between the depot to arc $v = a(j)$ and arc $u = a(i)$ to the depot respectively. Furthermore, let $q_i = q(u)$ denote the load and $c_i = c(u)$ denote the traversal cost of mapped arc $u = a(i)$ of node $i \in V'$.

While the algorithm of Vidal (2015) is specified for the VRP, it can be adjusted to account for arc routing. A cumulative traversal costs data structure $A[i]$ is constructed, where $A[i]$ represents the cumulative traversal costs of the mapped arcs for the nodes up to $i \in \{1, ..., n\}$. Finally, the cost of an arc $(i, j) \in A'$ can be expressed as $c(i, j)$, representing a route visiting the required arcs mapped to node $i+1$ until node j. Equations 7 till 9 denote the auxiliary data structures, while Equation 10 describes the cost of arc $(i, j) \in A'$.

$$D[i] = \sum_{k=1}^{i-1} d_{k,k+1} \tag{7}$$

$$Q[i] = \sum_{k=1}^{i} q_k \tag{8}$$

$$A[i] = \sum_{k=1}^{i} c_k \tag{9}$$

$$c(i,j) = \begin{cases} d_{0,i+1} + D[j] - D[i-1] + A[j] - A[i] + d_{j,0}, & \text{if } Q[j] - Q[i] \leq W \\ \infty, & \text{otherwise.} \end{cases} \tag{10}$$

The algorithm is depicted in Algorithm 2. The array $p[k][t]$ represents the least cost needed by using k vehicles ending at node t for $k \in \{1, ..., K\}$ and $t \in \{k, ..., n\}$. The queue Δ maintains a set of non-dominated predecessors nodes ranked by increasing costs. In the queue, the front element represent the least cost predecessor node, while the back of the queue contains a non-dominated node with the highest predecessor cost. The queue is a

Algorithm 2: Linear Split: Fleet limited to m vehicles (Vidal, 2015)

```
1  for k = 1 to m do
2      for t = 0 to n do
3          $p[k][t] = \infty$
4  $p[0][0] = 0$
5  for k = 0 to m-1 do
6      $clear(\Delta)$
7      $\Delta \leftarrow (k)$
8      for t = k+1 to n do
9          $p[k+1][t] = p[k][front(\Delta)] + c(front(\Delta), t)$
10         if $t < n$ then
11             while $dominates$(k,t,back($\Delta$) do
12                 popBack()
13             pushBack(t)
14         while $Q[t+1] - Q[front(\Delta)] > Q$ do
15             popFront()
```

double-ended queue such that access, add and remove elements at the front and back of the queue in constant time. In the outer loop (line 5), the number of available vehicles k is incrementally increased, while within the inner loop (line 8) t is increased to compute the least cost routes using k vehicles to reach the t-th node. For each k, the queue is initialized with the node k (line 7) as routes must contain at least one node. For instance if $k = 1$ and $t = 2$, then if a new vehicle is considered in line 9, the front of the queue (i.e., the best predecessor node) must be node 1. Otherwise, if the front would have been 0, the second vehicle would service the first and second node, while the first vehicle would service none. Thus, by initializing the queue with node k, it is ensured that all routes must have at least one arc. In line 9, the least cost predecessor node $i = front(\Delta)$ is extended towards t, where $c(i, t)$ is equal to the cost of creating a new route servicing the arcs mapped to nodes $i + 1$ to t.

The $dominates(i, j)$ function is utilized in the algorithm to maintain the non-dominated predecessor queue, ranked by increasing costs. The function returns TRUE if and only if node i dominates j as predecessor. Let $f(k + 1, i, x)$ be the cost of extending predecessor node i to $x \in \{i + 1, ..., n\}$ and $k = \{0, 1, ..., K - 1\}$. Thus,

$$f(k, i, x) = p[k][i] + c(i, x). \tag{11}$$

An arbitrary node i can only dominate j if $f(k, i, x) \leq f(k, j, x)$ and $i > j$. Clearly the cost of extending to node t from i must be lower or equal to the cost of extending from j (i.e., $f(k, i, x) \leq f(k, j, x)$). Furthermore, as $Q[i]$ is monotonically increasing due to $q(u) > 0\ \forall u \in R$, $i > j$ indicates that an extension from i consumes less capacity than an extension from j (i.e., $Q[x] - Q[i] < Q[x] - Q[j]$). Thus, if $i < j$ while $f(k, i, x) \leq f(k, j, x)$, i does not dominate j, as it is possible that i is unable to extent to $x + 1$ if for example $Q[x+1] - Q[i] > W$. Therefore, by substituting Equation 10 into Equation 11, the dominates function can be expressed as

$$dominates(k, i, j) = \begin{cases} p[k][i] + d_{\sigma,i+1} - D[i + 1] - A[i] \leq \\ p[k][j] + d_{\sigma,j+1} - D[j + 1] - A[j], \text{ if } i > j \\ FALSE, \text{ otherwise.} \end{cases} \tag{12}$$

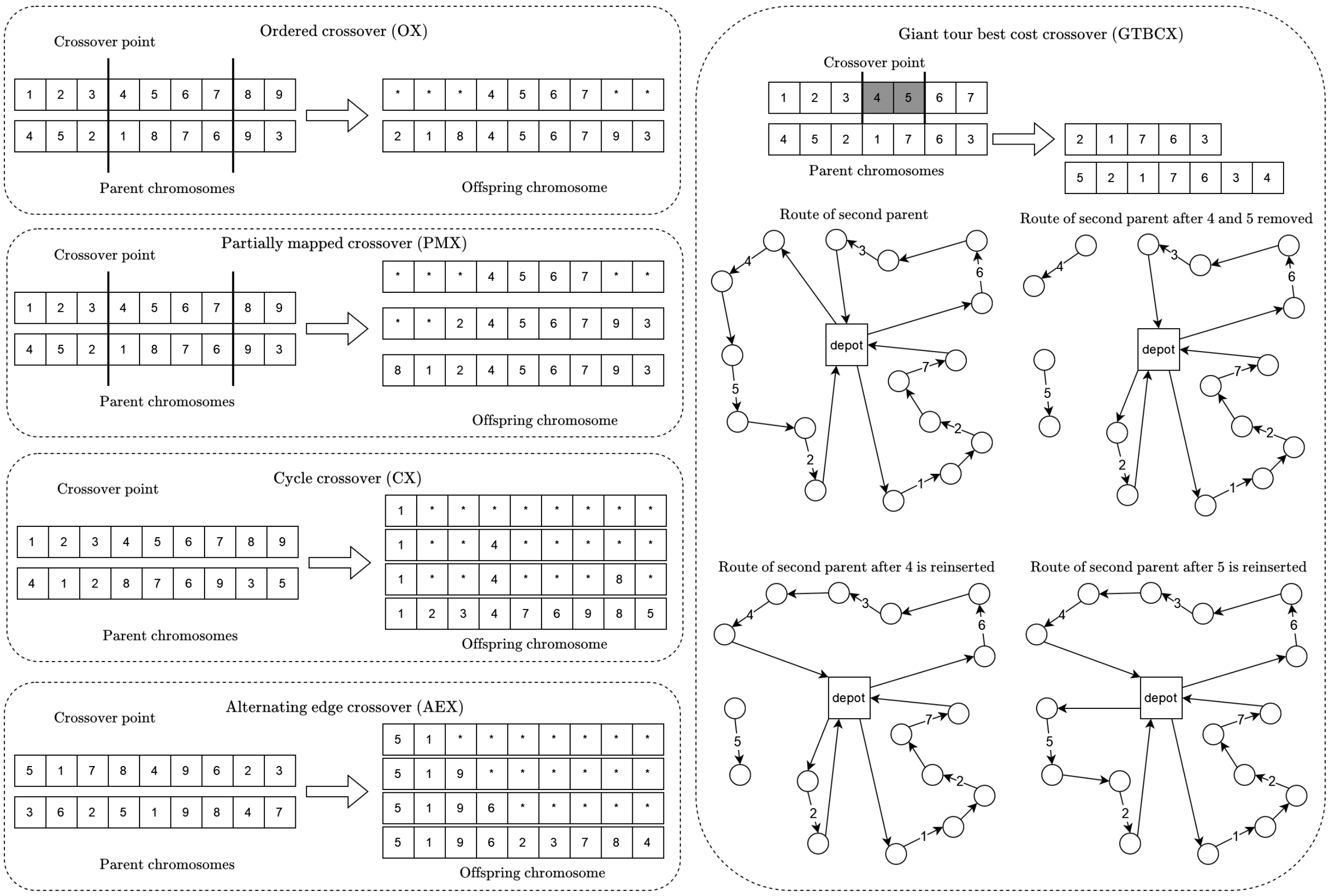

Ordered crossover (OX)
Crossover point
Parent chromosomes
Offspring chromosome
Partially mapped crossover (PMX)
Cycle crossover (CX)
Alternating edge crossover (AEX)
Giant tour best cost crossover (GTBCX)
Route of second parent
Route of second parent after 4 and 5 removed
Route of second parent after 4 is reinserted
Route of second parent after 5 is reinserted
depot

Figure 2: Crossover examples.

To ensure the non-dominated increasing cost queue, the front is pruned whenever the front node becomes infeasible due to capacity constraints (line 14). The back of the queue is pruned if the back node is dominated by a new predecessor (line 12) and the back of the queue is appended if a non-dominated predecessor is identified (line 13). The correctness of the algorithm is shown in Vidal (2015).

Crossover and mutation operators. As the chromosomes represent giant tours, crossovers designed for the Traveling Salesman Problem (TSP) can be utilized for the MA. Within the body of literature on the CARP, crossovers is a scarcely researched topic. Various studies research crossovers for TSP (Abdoun and Abouchabaka, 2012) and the VRP (Puljić and Manger, 2013). However, no such comparison has yet been provided for the CARP. Therefore, 8 crossovers are implemented: OX, PMX, CX, AEX, HGreX, GTBCX, MIX and a new crossover T-GTBCX. The methods of the crossovers are visually depicted in Figure 2. Furthermore, a mix of mutation operators are considered for their potential synergies with crossovers.

The ordered crossover (OX) copies a sequence arcs from the first parent into a offspring chromosome. Hereafter, the ordered arcs of the second parent complete the offspring. The sequence selected from the first parent is determined by randomly generating two points and selecting all arcs between these points. The selected sequence of arcs from the first parent is copied into the offspring at the same position and empty positions are filled with considered arcs starting from the second point of the second parent.

The partial mapped crossover (PMX) utilizes two randomly generated points. The section between the two points of the first parent copied into the offspring. Each arc in the section of the first parent is mapped to an arc at the same position in the second parent. Hereafter, the remaining positions are filled such that the positions of arcs in the second parent are adhered to as much as possible. Whenever an arcs cannot be positioned into the offspring, as it is already included in the offspring, its mapped counterpart is selected.

In the cycle crossover (CX), cycles are identified between the two parents. Where arcs for the offspring are selected from the first parent, while their positions are retrieved from the second parent. The cycle is started from the first arc in the first parent. Whenever the cycle is completed, all remaining positions are filled with arcs from the second parent.

The alternating edge crossover (AEX) selects successive arcs of the first and the second parent, based on the last selected arc. The procedure start with the first arc of the first parent and selects a random arc whenever a pair is infeasible.

The heuristic greedy crossover (HGreX) operates in a similar fashion as the AEX. However, pairs of successive arcs are not alternated between parents, but the cheapest successive pair is selected in terms of cost between the arcs.

The giant tour best cost crossover (GTBCX) was recently proposed by Sajid et al. (2021) for the VRP. The previously discussed crossovers discard any information on routes, and thus need to be evaluated after the construction of the offspring. The GTBCX applies a crossover operation on the routes rather than on the underlying chromosome. Therefore after creating a GTBCX offspring, the offspring does not need to be evaluated. In the GTBCX, two consecutive arcs of the chromosome are randomly selected from the first parent and these two arcs are removed from their respective routes of the second parent. Hereafter, the removed arcs are reinserted into the routes of the second parent at the cheapest cost position within these routes, where first the first removed arc is reinserted and hereafter the second arc is reinserted. This procedure of removing and reinserting an arc can be implemented in $\mathcal{O}(n)$ time for each arc. In this thesis, an additional extension is made to the GTBCX, the tabu-GTBCX (T-GTBCX). Within the T-GTBCX, two consecutive arcs are randomly selected from the first parent, and removed from their respective routes of the second parent. However, the routes of each respective removed arc are marked as tabu and each arc cannot be reinserted into their original route. Hereafter, the first removed arc is re-inserted into the least cost position obtained over all routes excluding its tabu route. Subsequently, the second arc is considered in the same manner. The GTBCX and T-GTBCX aims to place each arc into its optimal global position. However, the T-GTBCX could possibly enhance this procedure as the GTBCX can become trapped in sub-optimal regions.

The MIX crossover selects a random crossover OX, PMX, ERX, CX or AEX with equal probability. Finally, Four classical mutation operators are implemented. First the central inverse mutation (CIM), where a random point is selected. Hereafter, all arcs left and right of this point are inverted in the chromosome. Second the inverse mutation (IM), where the section between two random points are inverted within the chromosome. Third, the swap mutation (SM) swaps two random arcs. Finally, in the partial shuffle mutation (PSM), each arc is considered for swapping with another random arc with a probability. The mutation operators can be applied in conjunction with the GTBCX, as swaps of arcs (or sub-sequences) in the chromosome can be directly translated to swapping of arcs between routes while keeping the CARP solution intact.

Education. Each chromosome $\mathcal{S} \in \mathcal{P}$, produced within the MA, undergoes an education

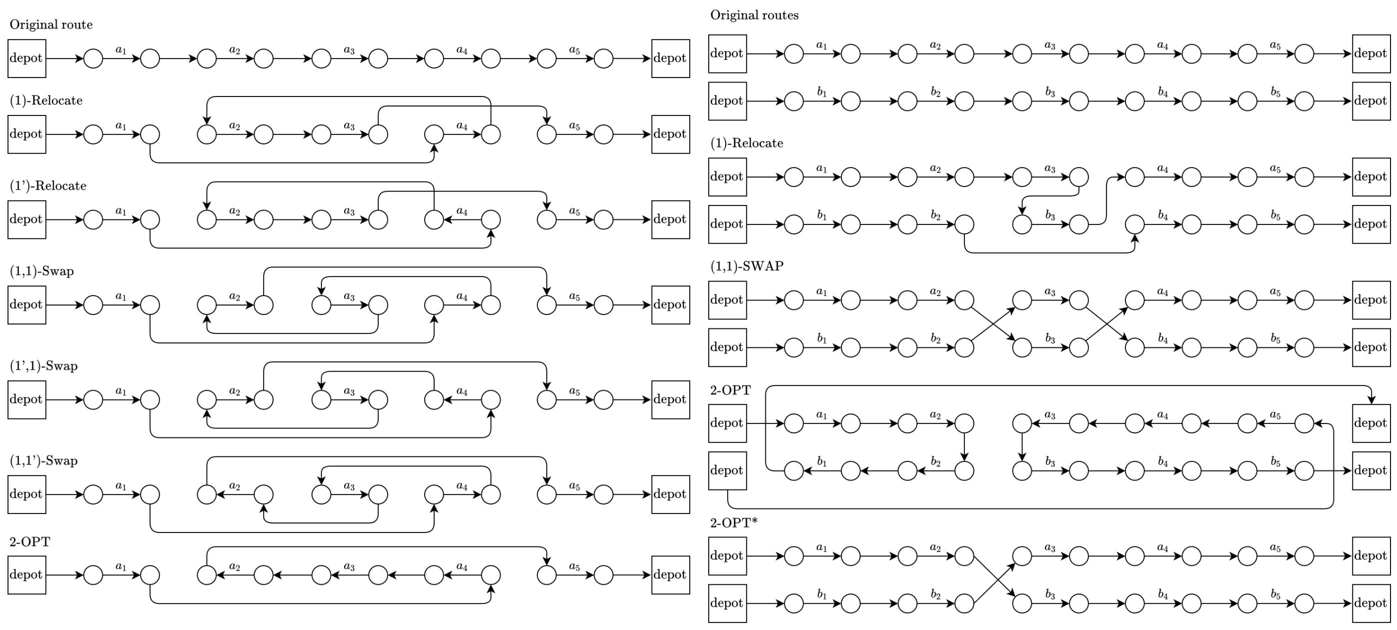


Figure 3: Selection of intra (left) and inter-route (right) moves.

procedure (e.g., local search). The procedure can only be called after the chromosome has been evaluated. Thus a CARP solution $\mathcal{T}(\mathcal{S})$ for $\mathcal{S}$ must have been produced. The education is tasked with improving the routes of the solution. The solution space of a solution includes relocate and swap moves, with possible reversals of the sequences up to 2 consecutive arcs, and finally 2-OPT and 2-OPT* moves (Laporte et al., 2014). A first improvement policy is implemented, whereby a move that decreases the total cost of the solution is directly applied. The education procedure terminates whenever no improving move could be identified while iterating over all arcs. All moves can be intra-route moves (e.g., within a single route) and inter-route moves. Figure 3 depicts a selection of these intra and inter-route moves. In a naive implementation of the education procedure, in each iteration, each arc evaluates all possible moves for all other arcs. Leading to a worst time-complexity of $\mathcal{O}(n^2)$ for each iteration of the education procedure. However, the solution space of a solution can be explored more efficiently, by utilizing neighbor lists for each arc $u \in R$ (Toth and Vigo, 2003). The neighborhood list $\Gamma(u) \subseteq R$ consists of $|\Gamma| \in \mathbb{Z}^+$ nearest arcs of u based on shortest path costs. The neighborhood lists reduce the worst-time complexity to $\mathcal{O}(n|\Gamma|)$, linear in the instance size.

Very recently, a new neighborhood SWAP* was proposed by Vidal (2022) for the capacitated VRP. The SWAP* swaps two customers between routes, while the standard swap move exchanges arcs in place, the SWAP* move exchanges arcs between routes without an insertion in place (i.e., one arc does not replace the other and vice-versa). The SWAP* is excellent at providing improvements at later stages of the local search when improvements are difficult to obtain (Vidal, 2022). A naive implementation of the SWAP* neighborhood runs in a time complexity of $\mathcal{O}(n^2)$. However, the neighborhood size can be reduced by only considering routes that are related to each other. In their study a polar coordinate overlap method is proposed. Where routes are only considered in the SWAP* search if their polar coordinates overlap. Thus greatly reducing the considered pairs of routes. However coordinates are not available in classical CARP instances. Thus to reduce the neighborhood size for these benchmark instances, where coordinate information is not available, a historical relatedness method is utilized (Arnold et al., 2021).

First, for each pair of arcs (u, v) where $u, v \in A$ and $u \neq v$, a global counter variable $z(u, v) \in \mathbb{Z}$ is used to count the number of times that arc u and v reside in the same route. Furthermore, as reverse arcs are present in G, $z(u, v) = z(u', v) = z(u, v') = z(u', v')$, where u' (v') is the reverse arc of u (v). The counter variable value serves as a proxy variable for the relatedness of arcs u and v. The counter variable $z(u, v)$ is updated whenever a new solution is introduced into the population in $\mathcal{O}(nb)$ average time complexity, where b is equal to the average number of arcs in routes.

Before the SWAP* search is performed, the relatedness between each route $\mathcal{T} \in \mathcal{T}(\mathcal{S})$ must be calculated, where $i = \{1, ..., |\mathcal{T}|\}$. Let $Z(\mathcal{T}^i, \mathcal{T}^j) \in \mathbb{Z}$ denote the relatedness between two routes, where $\mathcal{T}^i, \mathcal{T}^j \in \mathcal{T}(\mathcal{S})$, $\mathcal{T}^i \neq \mathcal{T}^j$ and $Z(\mathcal{T}^i, \mathcal{T}^j) = Z(\mathcal{T}^j, \mathcal{T}^i)$. Like the neighborhood list of arcs $\Gamma(a)$, each route $\mathcal{T} \in \mathcal{T}(\mathcal{S})$ is assigned a neighborhood list of routes $\Lambda(\mathcal{T})$ of size $|\Lambda|$ (i.e., the most $|\Lambda|$ related routes). The relatedness between routes can be calculated in $\mathcal{O}(n^2)$ if each arc in a route is compared to all other arcs of all other routes. However, this would not give any improvements in computational time as still all arc pairs are scanned. Thus a random arc sampling approach is implemented in this thesis. For each arc $u \in \mathcal{T}$ a set $C(u)$ of size $|\Lambda|b$ is constructed containing randomly sampled arcs from other routes, where b is the average number of arcs in each route. Let $t(u) \in \mathcal{T}$ denote the corresponding route in which arc u is serviced. Then, for each arc u and each $v \in C(u)$, $Z(\mathcal{T}^i, \mathcal{T}^j)$ is incremented by $\frac{z(u,v)}{|\mathcal{T}^i||\mathcal{T}^j|}$, where $\mathcal{T}^i = t(u)$ and $\mathcal{T}^j = t(v)$. The scalar $|\mathcal{T}^i||\mathcal{T}^j|$ scales the route relatedness by the number of possible combinations between pairs of arcs between route $\mathcal{T}^i$ and $\mathcal{T}^j$, as larger routes have a higher probability to be included in the random set $C(u)$ for each arc $u \in A$ due to the random procedure. The calculation of the route relatedness procedure runs in $\mathcal{O}(|\Lambda|Kb^2)$ time complexity, as for each arc in each route $|\Lambda|b$ other arcs are compared. Concluding, the original SWAP* procedure checks K^2 pairs of routes, while the historical relatedness SWAP* checks $K|\Lambda|$ pairs of routes. The original SWAP* runs in $\mathcal{O}(K^2b^2)$ and the historical related SWAP* in $\mathcal{O}(K|\Lambda|b^2)$.

Decomposition. Within the MA framework, a decomposition phase is called whenever the best solution has not been improved for it_{dec} iterations. Recall that a chromosome $\mathcal{S}$ has a corresponding CARP solution $\mathcal{T}(\mathcal{S}) = \{\mathcal{T}^1, ..., \mathcal{T}^{|\mathcal{T}|}\}$, and $\mathcal{T}$ can be converted back into a chromosome by route concatenation, where arcs within all routes are concatenated. Thus, to decompose the problem $\mathcal{S}$, routes of $\mathcal{T}(\mathcal{S})$ can be clustered and hereafter converted into a new chromosome to create a subproblem. Each subproblem can be solved independently and an improvement in one of the subproblems indicates that the original solution is improved. Each subproblem is solved by recursively calling the proposed MA framework.

Several strategies for decomposition within vehicle routing exist (Santini et al., 2021). However due to the lack of coordinates in instances, the historical relatedness $Z(\mathcal{T}^i, \mathcal{T}^j)$ between routes $\mathcal{T}^i$ and $\mathcal{T}^j$ can be utilized to cluster related routes together. A classical k-means clustering cannot be applied, as the historical relatedness cannot be projected onto a Euclidean space for all routes. Therefore, k-medoids clusters are constructed following the algorithm of Park and Jun (2009). In k-medoids clusters, the center of a cluster corresponds to an object (i.e., route) rather than a point in space. The k-medoids problem is NP-hard (Dupin and Talbi, 2018). Thus the problem is heuristically solved, and therefore does not guarantee an optimal cluster partition. An iteration of the k-medoids cluster algorithm can be implemented in $\mathcal{O}(kK^2)$, where k denotes the number of clusters and K the number of available vehicles. Each iteration, the k-medoids algorithm seeks to minimize the total relatedness across all clusters, by centers of clusters with new objects and hereafter reassigning objects to clusters. The k-medoids algorithm terminates if no improvement was found or the improvement was sufficiently small. The number of clusters, k, is based on the number of required arcs n within an instance. Santini et al. (2021) show that subproblems for routing of size 75 provided the best results, as problems of this size can be solved relatively quickly. Therefore, the number of clusters $k = \lceil \frac{n}{75} \rceil$.

4.2 Extensions to the CARP

Open-ended CARP. In order to solve the OECARP, the local search and the splitting procedure of the proposed MA framework must be modified. First, the local search can be easily adapted to omit the return distance of routes within the various moves (i.e., swap, relocate, 2-OPT, 2-OPT* and SWAP*). Thus the cost of route $\mathcal{T} \in \mathcal{T}(\mathcal{S})$ is redefined as

$$\operatorname{cost}(\mathcal{T}) = \operatorname{dist}(\sigma, tl(\mathcal{T}_1)) + \sum_{j=1}^{|\mathcal{T}|-1} [c(\mathcal{T}_j) + \operatorname{dist}(hd(\mathcal{T}_j), tl(\mathcal{T}_{j+1}))] \tag{13}$$

Second, in the split procedure, $d_{j,\sigma}$ is removed from Equation 10 as the return cost can be omitted. The cost of arc $(i, j) \in A'$ is reformulated as

$$c(i,j) = \begin{cases} d_{\sigma,i+1} + D[j] - D[i-1] + A[j] - A[i], \text{ if } Q[j] - Q[i] \leq W \\ \infty, \text{ otherwise.} \end{cases} \tag{14}$$

Other than a new cost function, the splitting procedure remains unchanged as other parts of the split procedure are independent of the return cost.

Multiple depot OECARP. To further extent the OECARP to allow routes to depart from multiple depots, the local search and split procedure are revised. To this extent, in the initialization of the algorithm for each arc $u \in R$ its nearest depot $\sigma(u) = \operatorname{argmin}_{\sigma \in D} dist(\sigma, tl(u))$ is precalculated, where $D \subseteq A$. The cost of a route $\mathcal{T}$ is adjusted to include the nearest depots for the first arc of a route $\mathcal{T} \in \mathcal{T}(\mathcal{S})$:

$$\operatorname{cost}(\mathcal{T}) = \operatorname{dist}(\sigma(\mathcal{T}_1), tl(\mathcal{T}_1)) + \sum_{j=1}^{|\mathcal{T}|-1} [c(\mathcal{T}_j) + \operatorname{dist}(hd(\mathcal{T}_j), tl(\mathcal{T}_{j+1}))] \tag{15}$$

The lookup for nearest depot for arc u can be implemented in $\mathcal{O}(1)$ time, thus the time complexity does not increase by including multiple depots in the OECARP local search component. The splitting procedure undergoes similar changes, where a set of depots is utilized. To this extent, the distance from the depot, $d_{\sigma,i+1}$, is replaced by the distance from the nearest depot of the arc mapped to node $i + 1$, denoted by $\sigma(i + 1)$ in Equation 10. Hence the cost of arc $(i, j) \in A'$ becomes

$$c(i,j) = \begin{cases} d_{\sigma(i+1),i+1} + D[j] - D[i-1] + A[j] - A[i], \text{ if } Q[j] - Q[i] \leq W \\ \infty, \text{ otherwise.} \end{cases} \tag{16}$$

Likewise, the dominance rule depicted in Equation 12 is adjusted to account for multiple depots:

$$dominates(i,j) = \begin{cases} p[k][i] + d_{\sigma(i+1),i+1} - D[i+1] - A[i] \leq \\ p[k][j] + d_{\sigma(j+1),j+1} - D[j+1] - A[j], \text{ if } i > j \\ FALSE, \text{ otherwise.} \end{cases} \tag{17}$$

As the lookup of $\sigma(u)$ is completed in $\mathcal{O}(1)$ for $u \in A$, no additional time complexity is introduced in the split procedure.

Heterogeneous MDOECARP. Finally, a fleet of heterogeneous vehicles is introduced. First, the local search procedure is adjusted to account for variable vehicle capacities W_t $\forall t \in T$. The heterogeneous capacities are included in the penalized cost of Equation 3. Thus the penalized cost for CARP solution $\mathcal{T}(\mathcal{S})$ becomes

$$\phi^{\text{COST}}(\mathcal{T}(\mathcal{S})) = \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{S})} \text{cost}(\mathcal{T}) + \delta^W \max[0, (\text{load}(\mathcal{T}) - W_{t(\mathcal{T})})] \tag{18}$$

where $t(\mathcal{T})$ denotes the vehicle type $t \in T$ for route $\mathcal{T} \in \mathcal{T}(\mathcal{S})$.

Second, a heterogeneous splitting procedure must be applied. To this extent, a label setting algorithm can be utilized for solving the procedure with heterogeneous vehicles. The problem of splitting a giant tour into routes with a heterogeneous fleet is equal to the Shortest Path Problem with Resource Constraints (SPPRC) (Prins, 2009). The SPPRC is proven to be an NP-hard problem (Dror, 1994). The SPPRC can be solved using a label setting dynamic programming approach (Desaulniers et al., 2005). This exact procedure is adopted in the heterogeneous vehicle CARP study by Prins (2009) and is implemented on the previously discussed auxiliary graph H.

In the approach of Prins (2009), the heterogeneous splitting procedure is tasked with constructing the least-cost path from the origin node 0 to the last node n in H. Furthermore, no more than a_k arcs (i.e., routes) can be assigned to vehicles of type $k \in T$. In each node $i \in V'$, multiple labels $L = (Z, \vec{x})$ can be saved. Where the cost Z of label L in node i denotes the incurred costs by utilizing $\vec{x} = \{0 \leq x_t \leq a_t, \forall t \in T\}$ vehicles to service all arcs mapped up to node i. The algorithm is initialized by creating a label $L = (0, \vec{0})$ with zero cost and null resource consumptions in the origin node. Hereafter, a breath-first-search is performed and each label is extended to all feasible nodes in H. A label L residing in node i can be extended to node j using vehicle $t \in T$ if $x_t + 1 \leq a_t$, $i < j$ and $Q[j] - Q[i] \leq W_t$. This procedure is repeated until all labels have been extended. The described label setting algorithm operates in $\mathcal{O}(n^2 \psi |T|)$ worst time-complexity, where $\psi = \prod_{k=1}^{|T|}(a_k + 1)$ denotes the number of possible permutations of the resource consumption labels, null consumptions included.

This process can be sped-up considerably by applying dominance and capacity checks (Prins et al., 2014). First, labels residing in the same node can be checked for dominance. A label $L = (Z, \vec{x})$ weakly dominates label $L' = (Z', \vec{x'})$ if $Z \leq Z'$ and $x_t \leq x'_t$ $\forall t \in T$. A weakly dominated label L is at least as good as L' in the CARP and thus L' can be removed. Second, capacity checks can be utilized. Let $F(\vec{x})$ be the residual capacity of the fleet of heterogeneous vehicles for resource consumption vector $\vec{x}$. If an extension to node j with vehicle type t using label $L = (Z, \vec{x})$ is considered and $Q[n] - Q[i] \leq F(\vec{x}) - W_t$, then a new label can be created. Otherwise if $Q[n] - Q[j] > F(\vec{x}) - W_t$, the extension towards j can be ignored as the remaining fleet does not contain enough capacity to service the arcs mapped to nodes $j + 1$ until n. Finally, the vehicle types can be sorted increasingly in their capacities, in order of capacity to stop the extension as soon as possible (Prins et al., 2014).

The label setting splitting procedure can be further improved by applying upper and lower bounds (Duhamel et al., 2012). For each node $i \in V'$ a lower bound can be computed that provides a least possible cost of extending node i to n. Moreover, an upper bound provides a cost on the problem that will never be exceeded. The lower and upper bounds are combined as follows, let label $L = (Z, \vec{x})$ reside in node j, then if $Z + LB[j] > UB$ label L can be discarded. This follows from the fact that further extending from the label L with lowest possible cost (i.e., $LB[j]$) would result in a cost higher than the upper bound. However, in the studies discussing bounding techniques for the splitting procedure of routing problems

Algorithm 3: New heterogeneous vehicle splitting procedure

```
1  Initialize $\mathcal{X}$, $A$, $Q$ and $D$
2  for t= 0 to $n$ do
3      for $\vec{x} \in \mathcal{X}$ do
4          $p[\vec{x}][t] = \infty$
5  $\Delta \leftarrow \mathcal{X}$
6  $p[\vec{0}][0] = 0$
7  while $\Delta \neq \emptyset$ do
8      $\vec{x} \leftarrow poll(\Delta)$
9      for $k = 0$ to $|T| - 1$ do
10         $\Lambda \leftarrow |\vec{x}|$
11         if $x_k + 1 > a_k$ then
12             CONTINUE
13         for $t = |\vec{x}| + 1$ to $n$ do
14             $\vec{y} = \vec{x}$
15             $y_k = x_k + 1$
16             $p[\vec{y}][t] = \min[p[\vec{y}][t], p[\vec{x}][t] + c(k,\text{front}(\Lambda), j)]$
17             if $t < n$ then
18                 while $|\Lambda| > 0$ and dominates($\vec{x}$,t,back($\Lambda$)) do
19                     $popBack()$
20                 $pushBack(t)$
21                 while $|\Lambda| > 0$ and $Q[t+1] - Q[\text{front}(\Lambda)] > Q_k$ do
22                     $popFront()$
```

by Duhamel et al. (2012) and Duhamel et al. (2011), no procedures were disclosed to obtain lower or upper bounds for the shortest path on the auxiliary graph H.

In this thesis two heuristics for the upper and lower bounds are utilized. First, the lower bounds for all nodes can be computed in $\mathcal{O}(n^2)$ by applying the unlimited homogeneous fleet splitting procedure of Vidal (2015) starting from each node $i \in V'$. The unlimited fleet splitting procedure runs in linear time, $\mathcal{O}(n)$. Furthermore, the capacity of vehicles is set to $W = \max_{\forall t \in T} W_t$, and thus the splitting procedure represents a relaxation of the original problem. As first the imposed fleet size is relaxed and second the capacities are equal to the largest capacity of the original fleet. Thus calling the linear split on each node $i \in V'$ provides a lower bound on the cost of the split from node i until n. For the upper bound, a simple greedy approach is applied. Starting from the smallest capacity vehicle, all nodes starting from $i = 0$ until j are assigned to that vehicle, such that its capacity is not violated. Hereafter, $i = j$ and the process is repeated for the next smallest vehicle until all vehicles have been used and the last node is visited. While this procedure runs in linear time complexity, it is not necessarily optimal and might not even return a feasible split.

New heterogeneous splitting procedure. However, in this thesis a new procedure is proposed to solve the heterogeneous splitting procedure in $\mathcal{O}(n\psi|T|)$ worst-time complexity. A factor n faster compared to the worst-time complexity of the label setting algorithm $\mathcal{O}(n^2\psi|T|)$. The new procedure is based on the previously discussed splitting procedure by Vidal (2015) with the introduction of heterogeneous vehicles. Algorithm 3 depicts the proposed splitting procedure.

First, the previously discussed data structures $Q[i], D[i]$ and $A[i]$ are initialized in $\mathcal{O}(n)$ based on the same auxiliary graph H (line 1). Hereafter, the array that contains the least cost path is denoted as $p[\vec{x}][t]$, where the vector $\vec{x} = \{0 \leq x_t \leq a_t, \forall t \in T\}$ denotes the number of utilized vehicles per type. Thus $p[\vec{x}][t]$ denotes the least cost path using $\vec{x}$ vehicles to reach node t of graph H. Let $\mathcal{X}$ be the set of all possible permutations of consumption vectors $\vec{x}$, then the size of $\mathcal{X}$ is equal to $\psi = \prod_{k=1}^{|T|}(a_k + 1)$. Second, each $p[\vec{x}][t]$ $\forall \vec{x} \in \mathcal{X}$ and $t \in V'$ must be initialized in $\mathcal{O}(\psi n)$ (line 4). Third, a linked-list Δ is filled with all permutations of the consumption vectors $\vec{x} \in \mathcal{X}$ (line 5). Within the list Δ, $\vec{x}$ must be preceded by $\vec{y} = \{y_i = x_i \ \forall i \in |T| \setminus j, y_j = x_j - 1\}$ if $x_j > 0$, $\forall j \in |T|$. For instance, (0,2,1) must be preceded by (0,2,0) and (0,1,1) and so forth. The initialization of the linked-list Δ can be integrated into the initialization of $p[\vec{x}][t]$ such that the time complexity remains $\mathcal{O}(\psi n)$. Finally, $p[\vec{0}][0]$ is set to 0 to conclude the initialization phase (line 7).

In the main algorithm, for each consumption vector $\vec{x} \in \Delta$ (line 8) a new route using vehicle of type $t \in T$ (line 9) is considered to each node $i \in V'$ (line 13), leading to a time complexity of $\mathcal{O}(n\psi|T|)$. Similarly to the split procedure of Vidal (2015), a double-linked queue Λ is maintained for each vehicle type $t \in T$ extended from $\vec{x}$ (line 10). The queue Λ is initialized with the total number of vehicles used in the consumption vector $\vec{x}$ (line 10), as every route must contain at least one arc. If an additional vehicle consumption of type $t \in T$ would violate the number of available vehicles a_t of type t (i.e., $x_t + 1 > a_t$), the extension of $\vec{x}$ using an addition vehicle of type t is terminated (line 11). Otherwise, a new route using vehicle type t extending towards each node $j \in V'$ is considered, where $j > |\vec{x}| = \sum_{i=1}^{|T|} x_i$. The cost of extending from node i towards j using a vehicle of type t is equal to

$$c(k, i, j) = \begin{cases} d_{0,i+1} + D[j] - D[i-1] + A[j] - A[i] + d_{j,0}, & \text{if } Q[j] - Q[i] \leq W_t \\ \infty, & \text{otherwise.} \end{cases} \tag{19}$$

The ordering within the linked list of vehicle consumption vetors Δ is of importance for setting the correct value of $p[\vec{x}][t]$. As the consumption vector $\vec{x}$ can be achieved by extending multiple vectors $\vec{y} = \{y_i = x_i \ \forall i \in |T| \setminus j, y_j = x_j - 1\}$ if $x_j > 0$, $\forall j \in |T|$. Therefore, the cost of $p[\vec{y}][i]$ must be set before $p[\vec{x}][j]$ can be considered, as the latter is dependent on the former. Furthermore, the cost attained by extending to node t using vehicle type t while previously using $\vec{x}$ vehicle types, is only saved if the evaluated extension improves the already saved cost in $p[\vec{x}][j]$ (line 16). This ensures that the least cost path in H to reach node j using consumption vector $\vec{x}$ is correctly saved. The queue Δ utilizes a dominance function, similar to the previous described dominance function in Equation 12, as to maintain a set of non-dominated predecessors ranked by increasing cost.

$$dominates(\vec{x}, i, j) = \begin{cases} p[\vec{x}][i] + d_{0,i+1} - D[i+1] - A[i] \leq \\ p[\vec{x}][j] + d_{0,j+1} - D[j+1] - A[j], \text{ if } i > j \\ FALSE, \text{ otherwise.} \end{cases} \tag{20}$$

The proof for the correctness of the algorithm follows the same procedure as described in Vidal (2015), as only a vector $\vec{x}$ for the number of used vehicles was introduced instead of a scalar k.

Adjustments of the heterogeneous splitting procedure for the OECARP or MDOECARP are relatively simple. For the OECARP, $d_{j,0}$ is removed from Equation 19, the cost of extending node i towards j. Similarly, for the MDOECARP, $d_{0,i+1}$ is substituted by $d_{d(i+1),i+1}$ in Equation 19 and 20.

Algorithm 4: NSGA framework of Lacomme et al. (2006)

```
1  $g \leftarrow 0$
2  Initialize subpopulations $\mathcal{P}_g$
3  Non-dominated sort and Assign crowding distances to $\mathcal{P}_g$
4  for $g < it_{max}$ do
5      while $|\mathcal{Q}_g| < \mu$ do
6          Apply binary tournament on $\mathcal{P}_g$ to obtain parents $P_1$ and $P_2$
7          Create offspring O from $P_1$ and $P_2$ by crossover
8          Evaluate O by splitting procedure
9          Improve O by local search procedure
10         Insert O into $\mathcal{Q}_g$
11     $\mathcal{R}_g = \mathcal{P}_g \cup \mathcal{Q}_g$
12     if $g = k \cdot it_{front}$ where $k \in \mathbb{N}$ then
13         Apply front improve $\mathcal{R}_g$
14         Non-dominated sort and Assign crowding distances to $\mathcal{R}_g$
15     Non-dominated sort and Assign crowding distances to $\mathcal{R}_g$
16     $\mathcal{P}_{g+1} \leftarrow$ update($\mathcal{R}_g$)
17     $g \leftarrow g + 1$
18 Return first front solutions
```

4.3 Multi-objective framework

To solve the Multi-Objective CARP (MO-CARP) a Multi-Objective Genetic Algorithm (MOGA) framework is implemented: the Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2002). The NSGA-II is closely related to the memetic algorithm discussed in Section 4.1, as solutions are encoded in chromosomes and other functions (e.g., crossover and local search) of the MA can be applied within the NSGA framework. In the following section, first the framework of the NSGA of Lacomme et al. (2006) for the CARP is described. Hereafter, extensions to the NSGA framework are presented based on recent developments in MAs for the CARP, as described in Section 4.1.

NSGA framework. In multiple-objective optimization problems, a solution is said to be Pareto-efficient if none of the objective values can be improved without degrading another objective (Borwein, 1983). The set of Pareto-efficient solutions represents the Pareto-front. The considered objectives of the framework of Lacomme et al. (2006) are the total cost and the makespan. The total cost is equal to the sum of all costs of its routes, defined by Equation 21 and the makespan of a solution $\mathcal{T}(\mathcal{S})$ is defined by Equation 22.

$$\text{total cost}(\mathcal{T}(\mathcal{S})) = \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{S})} \text{cost}(\mathcal{T}) \tag{21}$$

$$\text{makespan}(\mathcal{T}(\mathcal{S})) = \max_{\mathcal{T} \in \mathcal{T}(\mathcal{S})} [\text{cost}(\mathcal{T})] \tag{22}$$

For ease of notation, let $f_1(\mathcal{T}(\mathcal{S}))$ and $f_2(\mathcal{T}(\mathcal{S}))$ denote the makespan and total cost of $\mathcal{T}(\mathcal{S})$ respectively. Furthermore, a solution is said to be a Pareto-improvement if $\mathcal{T}(\mathcal{S})'$ dominates a solution $\mathcal{T}(\mathcal{S})$. A solution $\mathcal{T}(\mathcal{S})$ is dominated by $\mathcal{T}(\mathcal{S})'$ if

$$\begin{gathered} f_1(\mathcal{T}(\mathcal{S})') \leq f_1(\mathcal{T}(\mathcal{S})) \text{ and } f_2(\mathcal{T}(\mathcal{S})') < f_2(\mathcal{T}(\mathcal{S})) \\ \text{or } f_1(\mathcal{T}(\mathcal{S})') < f_1(\mathcal{T}(\mathcal{S})) \text{ and } f_2(\mathcal{T}(\mathcal{S})') \leq f_2(\mathcal{T}(\mathcal{S})). \end{gathered} \tag{23}$$

Otherwise, if a solution is not dominated by any other solution, the solution is said to be a non-dominated solution. The NSGA-II framework of Lacomme et al. (2006) is depicted in Algorithm 4. First, an initial population is generated by constructing μ chromosomes containing the required arcs in random order. Hereafter, a sorting and crowding distance assignment procedure is initiated. In the sorting procedure, all solutions are sorted into non-dominated fronts $\mathcal{F} = \{F_0, F_2, ..., F_{|\mathcal{F}|}\}$. Where the first front F_0 corresponds to all solutions not dominated by any other solution. Hereafter, the second front F_1 is constructed by removing the first front from $\mathcal{P}_g$ and subsequently sorting all remaining non-dominated solutions. This procedure is repeated until all chromosomes $\mathcal{S} \in \mathcal{P}_g$ are assigned to a front. After the non-dominated sorting, within each front $F \in \mathcal{F}$ each chromosome is assigned a crowding distance, representing the area of the largest cuboid enclosing a solution without the inclusion of other solutions within the front. Figure 4 depicts the non-dominated sorting and crowding distances. All fronts can be identified in at most $\mathcal{O}(|\mathcal{P}|^2 m)$ computations, where m is the number of objectives. Furthermore, the crowding distance measures for a front can be calculated in $\mathcal{O}(m|\mathcal{P}|\log|\mathcal{P}|)$ (Deb et al., 2002).

In the NSGA-II, successive populations are constructed until it_{max} generations have passed. Within each generation g, μ additional chromosomes are created from $\mathcal{P}_g$. To this extent, a binary tournament is applied to solutions in $\mathcal{P}_g$ for the parent selection. Candidates belonging to lower fronts are selected as parents. Whenever both candidates within the binary tournament reside in the same front, the candidate with the highest crowding-distance is selected as parent in order to contribute towards a diverse Pareto-front. Hereafter, an offspring is created by applying a crossover operator to the selected parents and subsequently the offspring is evaluated and hereafter educated by the local search procedure.

In the study by Lacomme et al. (2006), the authors note that applying a splitting procedure based on the total cost had no impact on the overall performance compared to only considering makespan. Therefore, the previously described $\mathcal{O}(nm)$ splitting procedure on total cost can be applied for n arcs and m vehicles. Furthermore, the local search applies a first improvement criteria, where an improvement is made if a new solution is Pareto-efficient (i.e., $\mathcal{T}(\mathcal{S})'$ dominates the previous solution $\mathcal{T}(\mathcal{S})$).

After the evaluation and education, the new offspring is included into a temporary population $\mathcal{Q}_g$ until $|\mathcal{P}_g| = \mu$. Hereafter, the two populations $\mathcal{P}_g$ and $\mathcal{Q}_g$ are merged into $\mathcal{R}_g$. Subsequently, $\mathcal{R}_g$ is non-dominated sorted into fronts and crowding-distances are assigned to each solution of each front. Hereafter, starting from the first front, the new population $\mathcal{P}_{g+1}$ is filled by iteratively adding the solutions belonging to the fronts of the merged population $\mathcal{R}_g$. This procedure is repeated until the size of $\mathcal{P}_{g+1}$ would exceed μ if the next considered front would be added. To complete the new population $\mathcal{P}_{g+1}$, the best crowding-distance solutions in the considered front are iteratively added to the new population until $|\mathcal{P}_{g+1}| = \mu$. Figure 5 depicts the population update procedure. Hereafter, the procedure for obtaining successive populations is repeated until a stopping criteria is met (i.e., max number of generations).

In the NSGA-II for the MO-CARP proposed by Lacomme et al. (2006) a directional local search procedure is proposed, depicted in Figure 6. In this local search procedure, a new solution is accepted if

$$w(f_1(\mathcal{T}(\mathcal{S})') - f_1(\mathcal{T}(\mathcal{S}))) + (1-w)(f_2(\mathcal{T}(\mathcal{S})') - f_2(\mathcal{T}(\mathcal{S}))) < 0. \tag{24}$$

Where the weight $w \in [0,1]$ specifies the direction of the local search. Let f_c^{min} and f_c^{max} denote the minimum and maximum value for the objective of the entire population for

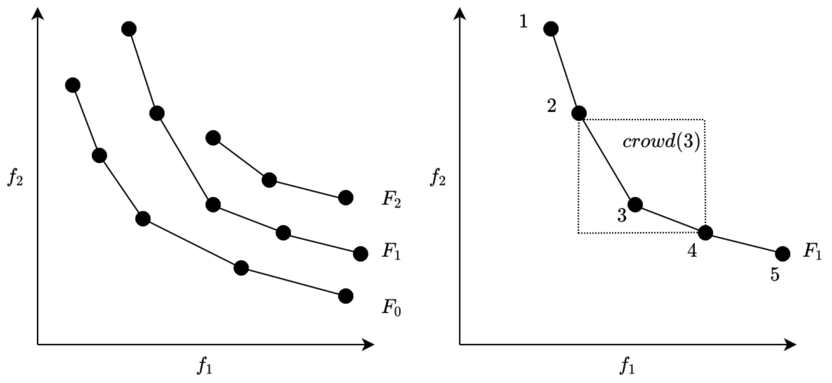


Figure 4: Non-dominated fronts (left) and crowding distance (right).

$c = 1, 2$. Then the weight w is defined as

$$w = \frac{\frac{f_1(\mathcal{T}(\mathcal{S})) - f_1^{min}}{f_1^{max} - f_1^{min}}}{\frac{f_1(\mathcal{T}(\mathcal{S})) - f_1^{min}}{f_1^{max} - f_1^{min}} + \frac{f_2(\mathcal{T}(\mathcal{S})) - f_2^{min}}{f_2^{max} - f_2^{min}}}. \tag{25}$$

Thus, the descent direction w resembles the objective improvement direction for a solution $\mathcal{T}(\mathcal{S})$. For example, if the solution value of $f_1(\mathcal{T}(\mathcal{S}))$ is relatively near f_1^{min} compared to $f_2(\mathcal{T}(\mathcal{S}))$ and f_2^{min}, then the improvement of solution $\mathcal{T}(\mathcal{S})$ places more emphasis on the improvement of f_1 (i.e., the makespan). Lacomme et al. (2006) obtained their best results by applying the directional local search every 10 generations on the entire population $\mathcal{R}_g$. Furthermore, to avoid early convergence in their MO-CARP framework, a clone management strategy was introduced. In a population $\mathcal{P}_g$, there can only be at most one solution with two given values (i.e., $\nexists\, \mathcal{S}_2 \in \mathcal{P}_g$ such that $f_1(\mathcal{T}(\mathcal{S})) = f_1(\mathcal{T}(\mathcal{S}_2))$ and $f_2(\mathcal{T}(\mathcal{S})) = f_2(\mathcal{T}(\mathcal{S}_2))$, $\forall \mathcal{S} \in \mathcal{P}_g$).

Improved NSGA-II framework for the CARP. To enhance the NSGA-II, recent developments of the MA for the single-objective CARP are included into the NSGA-II framework, as discussed in Section 4.1. Algorithm 5 depicts the proposed NSGA framework.

First, the population $\mathcal{P}$ is divided into two subpopulations: a feasible $\mathcal{P}^f$ and infeasible $\mathcal{P}^{inf}$ subpopulation. To this extent, an infeasibility penalty factor is included in the objective functions. The makespan is redefined as

$$f_1(\mathcal{T}(\mathcal{S})) = \max_{\mathcal{T} \in \mathcal{T}(\mathcal{S})} (\text{cost}(\mathcal{T}) + \delta^W \max[0, (\text{load}(\mathcal{T}) - W)]) \tag{26}$$

and total cost as $f_2(\mathcal{T}(\mathcal{S})) = \phi^{\text{COST}}(\mathcal{T}(\mathcal{S}))$. Solutions within the NSGA-II can therefore be included into the correct subpopulation based on its feasibility. In each generation, μ chromosomes are created, evaluated, educated and included into the corresponding subpopulation. The binary tournament selection is adjust to account for infeasible solutions. A candidate solution is selected as parent if it dominates the other candidate. Whenever both candidates are non-dominating, the candidates with the largest crowding-distance is selected as parent. Furthermore, whenever the resulting offspring is infeasible, an effort to repair the offspring is exerted with probability p_{REP}.

A diversification phase is initiated whenever the NSGA-II could not improve the first front F_0 of feasible solutions for it_{div} number of generations. In the diversification phase, each merged subpopulation $\mathcal{R}_g^i$ is pruned down to size $\min(\frac{\mu}{3}, |F_0|)$, where $|F_0|$ denotes the size

Algorithm 5: Proposed NSGA framework for the MO-CARP

```
1  $g \leftarrow 0$
2  Initialize subpopulations $\mathcal{P}_g^f$ and $\mathcal{P}_g^{inf}$
3  Non-dominated sort($\mathcal{P}_g^f$) and Non-dominated sort($\mathcal{P}_g^{inf}$)
4  Assign crowding distances($\mathcal{P}_g^f$) and Assign crowding distances($\mathcal{P}_g^{inf}$)
5  while number of iterations without improvement $\leq it_{max}$ and time $\leq T_{max}$ do
6      while $|\mathcal{Q}_g^{inf}| + |\mathcal{Q}_g^{inf}| < \mu$ do
7          Apply binary tournament on $\mathcal{P}_g^f \cup \mathcal{P}_g^{inf}$ to obtain parents $P_1$ and $P_2$
8          Create offspring O from $P_1$ and $P_2$ by crossover
9          Evaluate O by splitting procedure
10         Improve O by local search procedure
11         Adjust penalty parameter
12         if (O is infeasible) then
13             Insert O into $\mathcal{Q}_g^{inf}$
14             Repair O with probability $P_{REP}$
15         else
16             Insert O into $\mathcal{Q}_g^f$
17     $\mathcal{R}_g^f = \mathcal{P}_g^f \cup \mathcal{Q}_g^f$ and $\mathcal{R}_g^{inf} = \mathcal{P}_g^{inf} \cup \mathcal{Q}_g^{inf}$
18     Non-dominated sort($\mathcal{R}_g^f$) and Non-dominated sort($\mathcal{R}_g^{inf}$)
19     Assign crowding distances($\mathcal{R}_g^f$) and Assign crowding distances($\mathcal{R}_g^{inf}$)
20     if number of iterations without improvement $= k \cdot it_{front}$, where $k \in \mathbb{N}$ then
21         Apply front improve search on $\mathcal{R}_g^f$ and $\mathcal{R}_g^{inf}$
22         Non-dominated sort($\mathcal{R}_g^f$) and Non-dominated sort($\mathcal{R}_g^{inf}$)
23         Assign crowding distances($\mathcal{R}_g^f$) and Assign crowding distances($\mathcal{R}_g^{inf}$)
24     if number of iterations without improvement $= k \cdot it_{div}$, where $k \in \mathbb{N}$ then
25         Remove solutions from $\mathcal{R}_g^f$ and $\mathcal{R}_g^{inf}$
26         Initialize $\mu$ new chromosomes
27         Non-dominated sort($\mathcal{R}_g^f$) and Non-dominated sort($\mathcal{R}_g^{inf}$)
28         Assign crowding distances($\mathcal{R}_g^f$) and Assign crowding distances($\mathcal{R}_g^{inf}$)
29     $\mathcal{P}_{g+1}^f \leftarrow$ update($\mathcal{R}_g^f$)
30     $\mathcal{P}_{g+1}^{inf} \leftarrow$ update($\mathcal{R}_g^{inf}$)
31     $g \leftarrow g + 1$
32 Return first front solutions
```

of the first front of the corresponding subpopulation $\mathcal{R}_g^i$. In the survival procedure, fronts starting from the last front are iteratively removed from the subpopulation until the removal of the next considered front would remove too many solutions. From this considered front, solutions with the highest crowding distance are kept, such that $|\mathcal{R}_g^i| = \min(\frac{\mu}{3}, |F_0|)$, while others are discarded.

A decomposition phase is introduced into the NSGA-II after no improvement is identified on the first front in it_{dec} generations. In this phase, a random solution belonging to the first front of either subpopulation is selected. Hereafter, the solution is decomposed into subproblems and the local search procedures of each subproblem is guided by the direction weight w. Where w, defined in Equation 25, is based on the selected random solution.

Finally, a new clone management procedure is proposed in this thesis. In the original NSGA-II framework for the MO-CARP, solutions with identical solution costs would not be included into a subpopulation. However, if only unique solutions are permitted, the NSGA-II might have difficulty improving the first front if this front has few solutions. Thus,

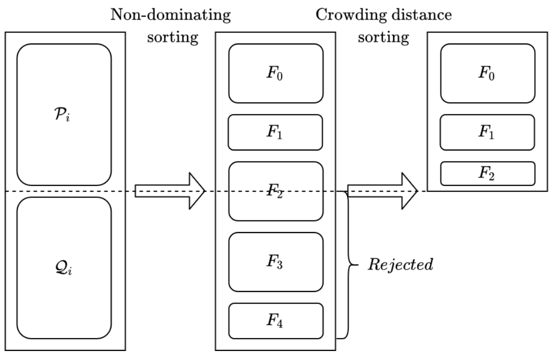


Figure 5: Population update.

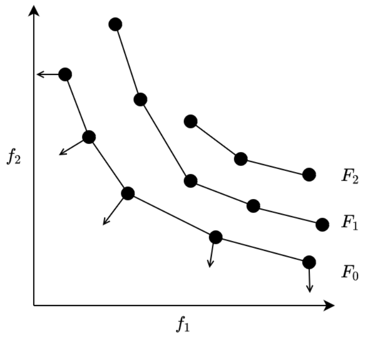


Figure 6: Front improvement.

if duplicated solutions are allowed, the duplicated solutions on the first front may serve to better explore the front. However, by allowing these duplicates, it is possible that the first front clusters on a single value, instead of a uniform spread on the front. To counter this, a restriction is set on the number of solutions that can cluster on a single point. This number of allowed duplicate solution values is set equal to $\frac{\mu}{|F_0|}$, to ensure that each point on the first front may have an equal number of duplicates and therefore the first front converges to a uniform distribution of duplicates. Limiting the number of duplicates be achieved by removing duplicates when solutions of a merged subpopulation $\mathcal{Q}_g$ are selected for its successor subpopulation $\mathcal{P}_{g+1}$. In this selection procedure, for each cluster of duplicate solutions where the number of duplicates exceeds $\frac{\mu}{|F_0|}$, the worst duplicates in terms of their broken-pairs distances are iteratively removed until $\frac{\mu}{|F_0|}$ solutions remain in the cluster. Allowing these duplicates could potentially permit the NSGA-II framework to examine the first front more intensively, while maintaining diversity between duplicates due to their broken-pair distances.

5 Computational experiments

Extensive experiments are performed to investigate the proposed contributions in this thesis. First, the heterogeneous splitting procedure is analyzed. Second, the analysis of the effectiveness of classical and the new T-GTBCX crossover operators are presented. Third, the results of the MA for the CARP on benchmark instances are provided. Hereafter, the classical CARP instances are extended to the OECARP, MDOECARP, HCARP, HOECARP and the HMDOECARP. Finally, the results obtained by the proposed NSGA-II framework for the MO-CARP are presented.

All algorithms were implemented in the Java programming language (version 8). The experimental results are obtained using Windows 10 with an Intel i5-4670K processor (4 cores at 3.8GHz) and 16GB RAM. All obtained results within this thesis for the ARPs can be downloaded from Github.

5.1 Heterogeneous splitting procedure

A set of 96 real-world routing instances (Duhamel et al., 2010), containing between 20 and 256 customers and a single depot, are used for the comparison of the splitting procedures. Within each instance, each customer is assigned a random demand between 1 and 50. Hereafter, a giant tour is produced for each instance using the extended Lin-Kernighan-Helsgaun (LKH-3) solver for traveling salesman problems (Helsgaun, 2017). The proposed heterogeneous splitting procedure (Algorithm 3) is compared to the label setting algorithm of Prins (2009) on the produced giant tours of each instance. The label setting algorithm utilizes the discussed speed-up techniques of Section 4, such as lower and upper bounds.

The splitting procedures could be simply compared on existing heterogeneous vehicle routing instances. However a more in-depth analysis can be made by analyzing the effect of the capacity distribution of the fleet on the distribution of the vehicles and the number of heterogeneous vehicle types. To this extent, 3 variations of capacity distributions are considered for each instance. A relative difference of 25%, 50% and 75% of capacities for subsequent vehicle types. Let $\vec{c}$ denote the capacities for a fleet of $|T|$ vehicle types, where $c_i > 0 \ \forall i = \{0, 1, ..., |T| - 1\}$ denotes the capacity for each vehicle in the fleet. The capacity of the first vehicle is denoted by C and subsequent capacities follow the relative differences $\beta = \{25\%, 50\%, 75\%\}$. Thus, $c_0 = C$ and $c_i = c_{i-1}(1 + \beta) \ \forall i = \{1, 2, ..., |T| - 1\}$. Furthermore, 3 distributions of the number of vehicles per vehicle type are considered. A linear decreasing, uniform and a linear increasing vehicle availability for subsequent vehicles within the fleet. More specifically, within the decreasing (increasing) vehicle availability, the number of available vehicles for each subsequent type is incrementally decreased (increased) by 3 vehicles. Let $\vec{a}$ denote the number of vehicles for each vehicle type, where $a_i > 0 \ \forall i = \{1, 2, ..., |T| - 1\}$ denotes the number of vehicles of type $i \in T$. The number of vehicles of the first vehicle type is equal to A, and subsequent number of vehicles for vehicle types are equal to $a_i = a_{i-1} + \alpha \ \forall i = \{1, 2, ..., |T| - 1\}$ for distribution types $\alpha = \{-3, 0, 3\}$. For instance, $\alpha = -3$ denotes a linearly decreasing fleet availability by steps of 3 (e.g., a fleet with 3 vehicle types and $A = 8$ would produce a fleet availability $\vec{a} = (8, 5, 2)$).

The capacity and fleet composition variations are analyzed for fleets consisting of $2, 3$ and 4 vehicle types. Resulting in a total of 27 parameter settings for each instance, thus creating a total of 2592 experiments $(3 \cdot 3 \cdot 3 \cdot 96)$. Values for C and A are determined for each instance, fleet distribution α and number of vehicle types $|T|$. The values of C and A are chosen such that the fleet resembles a realistic fleet. Therefore, the total capacity of the constructed fleet should not exceed the total demand by more than twice the capacity of the smallest vehicle type (Taillard, 1999). Hereafter, the experiments provide insights on the effects of increasing the relative capacities β for for a given C and A.

Each experiment is repeated 30 times to obtain the average speed-up factors over all 30 runs. Table 1 depicts the average speed-up factor results of the proposed heterogeneous splitting procedure over all instances for each parameter setting compared to the label setting algorithm, where rows represent the fleet distribution and columns the capacity distribution. On average, the largest speed-ups are measured on instances with a uniform fleet $\alpha = 0$ and small differences in vehicle capacities $\beta = 25\%$. Furthermore, the results show that for a given fleet composition, the speed-up decreases when the relative capacities of the fleet increased. The largest decrease in speed-ups is observed in fleets that consist of relatively many large vehicles ($\alpha = 3$). While the smallest decrease in speed-up is observed in fleets with relatively many small vehicles ($\alpha = -3$). This effect can be explained by the impact of increasing capacities, as increasing capacities on a fleet consisting of mainly large vehicles benefits the dominance checks within the label setting algorithm more than increasing capacities on a fleet consisting of mainly smaller vehicles.

Figure 7 depicts the speed-up results for $|T| = 4$ and $\alpha = -3$ for $\beta \in \{25\%, 50\%, 75\%\}$. In the figure, the effect of the instance size on the speed-up can be observed. As larger instances yield on average larger speed-up. Furthermore, the decrease in speed-up can be clearly observed across the 3 panels. Indicating the effectiveness of the proposed splitting procedure for larger instances or when capacity differences are relatively small.

Over all experiments, the largest observed speed-up factor was 121. However, speed-up

$\|T\| = 2$		β 25%	50%	75%
	-3	4.60	4.52	4.41
α	0	4.88	4.87	4.63
	3	3.45	3.26	3.09

$\|T\| = 3$		β 25%	50%	75%
	-3	4.48	4.38	3.31
α	0	5.17	4.62	3.24
	3	5.09	4.17	2.72

$\|T\| = 4$		β 25%	50%	75%
	-3	19.89	15.64	5.26
α	0	25.01	15.70	3.32
	3	24.64	14.78	2.70

Table 1: Average speed-up factor of proposed split procedure compared to label setting algorithm.

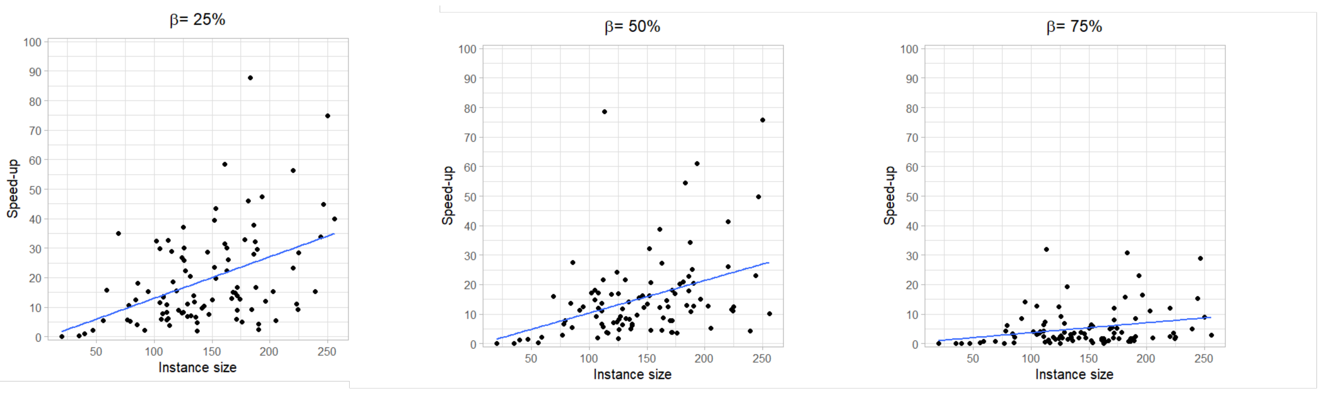


Figure 7: Speed-up factor differences between various relative capacity factors.

factors lower than 1 (i.e., the label setting algorithm was faster) were observed within the experiments. For example, for the experiments $|T| = 4$, $\beta = 0.25\%$ and $\alpha = \{-3, 0, 3\}$, only 2.4% of experiments resulted in a speed-up factor lower than 1. These experiments, with a speed-up factor lower than 1, were all smaller instances ($n \leq 40$) and the smallest observed speed-up factor for these experiments was 0.12. However, with a larger relative capacity difference $\beta = 50\%$ for $|T| = 4$ and $\alpha = \{-3, 0, 3\}$, 4.2% of the experiments resulted in a speed-up factor lower than 1. Where the instance sizes of experiments with speed-up factors smaller than 1 were smaller or equal to 78 (i.e., $n \leq 78$) and the smallest speed-up factor observed was equal to 0.03. Finally, for the largest relative differences in capacities $\beta = 75\%$, 29.9% of the experiments resulted in speed-up factors smaller than 1. The smallest speed-up factor was equal to 0.003 and the the instance sizes were smaller or equal to 190 (i.e., $n \leq 190$). Similar results are obtained for other values of $|T|$. Thus, the heterogeneous splitting procedure performs relatively well on larger instances were the fleet compositions does not exceed the required capacity by a wide margin (i.e., $\beta = \{25\%, 50\%\}$).

5.2 Crossover evaluation for the CARP

The experiments of the crossover comparison are based on a simplified version of the proposed MA framework in this thesis. The simplified version is depicted in Algorithm 6, where the decomposition and diversification phase are omitted, such that the effectiveness of the crossover operators can be compared without any external influence of the decomposition and the diversification phases within the memetic algoirthm framework. Eight variations of crossovers are tested. The OX, PMX, CX, AEX, MIX, HGreX, GTBCX and new crossover T-GTBCX, as described in Section 4. Each crossover operator is tested by applying the discussed mix of mutation operators with the following probabilities: 0%, 1%, 5% and 10% after the crossover has been performed. These combinations of crossover operators and mutation probability variations, result in a total of 32 variants. All 32 variants are tested on 4 instances of the classical EGL CARP instances set of Eglese (Eglese and Li, 1996). The selected instances from the EGL set represent the 4 largest instances of this set and

Algorithm 6: Memetic algorithm

```
1  Initialize subpopulation $\mathcal{P}_f$ and $\mathcal{P}_{inf}$
2  while number of iterations $\leq it_{max}$ do
3      Apply binary tournament to obtain parents $P_1$ and $P_2$
4      Create offspring O from $P_1$ and $P_2$ by crossover
5      Mutate O with probability $p_m$
6      Evaluate O by splitting procedure
7      Improve O by local search procedure
8      Adjust penalty parameter
9      if (O is infeasible) then
10         Insert O into $\mathcal{P}_{inf}$
11         Repair O with probability $P_{REP}$
12     else
13         Insert O into $\mathcal{P}_f$
14     if ($|\mathcal{P}_i| = \mu + \lambda$, where $i = f, inf$) then
15         Prune subpopulation $\mathcal{P}_i$ to size $\mu$
16 Return best feasible solution
```

Instance	Number of arcs	Number of vehicles	Best known solution value	Running time GTBCX based crossovers (s)	Running time other crossovers (s)
e4-C	98	19	11529	62.61	94.19
s2-A	147	14	9868	93.97	156.19
s3-A	159	15	10201	101.61	168.29
s4-A	190	19	12216	127.78	230.86

Table 2: Instance description.

represent a real-world street network. These instance contain 98 to 190 arcs, further descriptions of the EGL set is described in Table 2. Thus, a total of 128 experiments are considered for the crossover comparison.

To obtain the results for the comparison of crossovers, each experiment is repeated 30 times to account for the stochasticity present in the MA. The MA is terminated after $it_{max} = 100,000$ crossover evaluations, in order to evaluate the effectiveness of each crossover and mutation probablity combination. The parameter settings for the simplified MA use the original settings of Vidal (2022). The minimum subpopulation size $\mu = 20$, maximum subpopulation size $\mu + \lambda = 45$, neighborhood size of each arc $|\Gamma| = 20$, neighborhood size of routes for the SWAP* $|\Lambda| = 3$, number of elite solutions $n_{ELITE} = 4$, number of chromosomes considered for broken pairs distance $n_{CLOSEST} = 5$ and feasibility ratio $\xi^W = 0.2$.

The average running times of the GTBCX based crossovers are depicted in Table 2, as well as the average running times of all other crossovers. The averages of the best obtained solution values over 30 repeated runs for each crossover operator and instance is depicted in Table 3. The results for the crossover comparison with various mutation probabilities are shown in Tables 7, 8 and 9 in the Appendix. The best average solutions over all experiments are marked in bold for each instance. Furthermore, the gap is computed as $100(z - z_{BKS})/z_{BKS}$, where z is the average solution value and z_{BKS} denotes the best known solution value currently known for the instance.

	Crossover operator							
Instance	OX	HGreX	CX	AEX	PMX	MIX	GTBCX	T-GTBCX
e4-C	11736.9	11912.5	11615.3	11954.6	11626.6	11719.3	12138.9	**11591.4**
	1.80%	3.33%	0.75%	3.69%	0.85%	1.65%	5.29%	0.54%
s2-A	10082.4	10113.5	9978.8	10194.1	9996.5	10059.0	10157.6	**9895.9**
	2.17%	2.49%	1.12%	3.31%	1.30%	1.94%	2.93%	0.28%
s3-A	10354.5	10404.1	10285.8	10549.1	10319.4	10369.7	10518.0	**10247.3**
	1.50%	1.99%	0.83%	3.41%	1.16%	1.65%	3.11%	0.45%
s4-A	12531.0	12626.4	12434.0	12776.8	12444.5	12537.5	12782.2	12302.6
	2.58%	3.36%	1.78%	4.59%	1.87%	2.63%	4.63%	0.71%
Average	2.01%	2.79%	1.12%	3.75%	1.29%	1.97%	3.99%	**0.50%**

Table 3: Crossover comparison without mutation, percentages denote the gap$_{BKS}$.

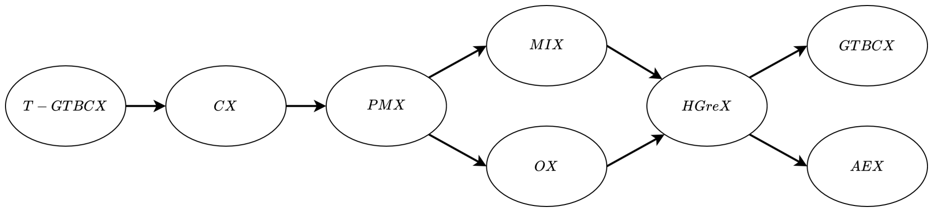


Figure 8: Statistical results on the crossover operators without mutation, depicting significant crossover operator superiority.

From the results, mutations did not always improve the performance of the MA, contrary to the findings of Puljić and Manger (2013) for the VRP. Furthermore, in their study, the HGreX was found to be an excellent crossover operator. However within the obtained results in this thesis, the HGreX is in the bottom 3 in terms of solution values. The recently proposed GTCBX crossover (Sajid et al., 2021) is the worst performing crossover of all considered crossovers. However, the novel T-GTBCX, proposed in this thesis, is the best crossover operator compared to all other crossovers over all instances. Furthermore, GTBCX based crossovers performed better in the running time (see Table 2), as the splitting procedure is bypassed due to the route information preservation within the GTBCX based crossovers. These results are further verified by statistical tests. First, for each instance, the differences of average solutions values between crossover operators are tested using the Student's t-test on the results of the crossovers without mutations. Second, for each instance, the differences in solution values for each crossover operator across the various mutation probabilities are tested by the ANOVA test.

The outcomes of the t-tests are visualized in Figure 8. Crossovers that showed a significant superiority over another crossover in all 4 instances are depicted on the left of its inferior operators. Furthermore, the difference in means of the various mutation probabilities for each crossover and each instance are tested by the ANOVA test, depicted in Table 10 in the Appendix. The ANOVA tests the null hypothesis that there exists no difference among the solutions for each crossover operator and instance over all mutation variations. The ANOVA tests showed that only twice a significant difference in the group means was observed. First, the CX operator and second the GTBCX operator were both found to be significantly decreasing in solution value when a 5% mutation probability was introduced. All other ANOVA tests showed no significant difference in group means. All mentioned statistical tests use a significance level of 5%.

5.3 Results for the CARP and its extensions

The best performing operator T-GTBCX is further analyzed on a larger set of classical CARP benchmark instances. To this extent, the T-GTBCX is incorporated in the proposed MA (Algorithm 3) to analyze its ability to produce improvements at later stages in the search. As in the previous subsections, the effectiveness of crossovers were analyzed for a limited number of generations. Thus, while the T-GTBCX could be superior in the first 100.000 crossovers, other crossovers could potentially obtain better results in the later stages of the MA. Therefore, the T-GTBCX is further compared to the commonly applied OX operator within the proposed MA. Finally, a current state-of-the-art MA for the CARP of Vidal (2017) is included for a final comparison.

Within the CARP literature, 5 sets of instances are commonly analyzed for the comparison of algorithms. In the performed experiments, only 2 instance sets are selected to further analyze the T-GTBCX: the EGL and EGL-L sets. The EGL (Eglese and Li, 1996) and EGL-L (Brandão and Eglese, 2008) instances are based on real-world street networks and represent the largest and most difficult CARP instances. Other CARP instances can be solved to optimality with relative ease with current state-of-the-art algorithms, therefore these instances are not considered for the experiments on the CARP. The characteristics of all instances are further provided in Table 13 in the Appendix.

Experimental results are obtained by applying the algorithm with a time limit of 60 minutes. All results are achieved by repeating the algorithm 10 times to account for the stochasticity present in the MA. The parameter settings of the proposed MA use the original settings of Vidal (2022). The MA is terminated after $it_{max} = 10,000$ generations have passed without any improvement, the minimum subpopulation size $\mu = 20$, maximum subpopulation size $\mu + \lambda = 45$, neighborhood size of each arc $|\Gamma| = 20$, neighborhood size of routes for the SWAP* $|\Lambda| = 3$, number of elite solutions $n_{ELITE} = 4$, number of chromosomes considered for broken pairs distance $n_{CLOSEST} = 5$ and feasibility ratio $\xi^W = 0.2$. A diversification phase is initiated when the MA could not improve the best solution value each $it_{div} = 4,000$ generations. Finally, the decomposition phase is performed each $it_{dec} = 2,000$ generations, counted from the last generation where an improvement was determined.

Table 4 depicts the results obtained by applying the T-GTBCX and OX operator in the developed framework, as well as the results of the MA of Vidal (2017). The first 24 instances represent the EGL instance set, while the last 10 instances come from the more difficult EGL-L set. The Wilcoxon signed-rank test is used to compare the gaps of the best obtained solutions over all 10 runs for each instance set. The T-GTBCX significantly produced better results compared to the OX in the MA framework, as the T-GTBCX and OX resulted in a gap of 0.07% and 0.24% respectively (p-value < 0.001). The T-GTBCX produced 21 of the 34 best known solutions (BKS), while the OX only provided 17 of the 34 BKS. Moreover, in 6 of the 34 from the EGL and EGL-L instances, new BKS were obtained by the T-GTBCX, depicted by the underline in the table. The old BKS and the improvements are depicted in Table 13 in the Appendix. Finally, the running time of the MA was improved by omitting the split procedure due to the T-GTBCX operator. On average the T-GTBCX ran roughly 33% faster on the EGL instances, while on the EGL-L set this further increased to 57%. Similar results are obtained when comparing the T-GTBCX to the results of the MA of Vidal (2017). A significant improvement was observed in terms of the average gap, 0.07% and 0.10% respectively (p-value $= 0.03$). Furthermore, a substantial difference in running time was observed between the T-GTBCX and the MA of Vidal (2017), similar to the previous discussed running time results of the T-GTBCX.

	OX			T-GTBCX			Vidal (2017)			
Instance	Average	Best	T(min)	Average	Best	T(min)	Average	Best	T(min)	BKS
e1-a	3,548	**3,548**	0.59	3,548	**3,548**	0.47	3,548	**3,548**	0.76	3,548
e1-b	4,498	**4,498**	0.55	4,498	**4,498**	0.44	4,498	**4,498**	1.13	4,498
e1-c	5,595	**5,595**	0.53	5,595	**5,595**	0.50	5,595	**5,595**	0.78	5,595
e2-a	5,018	**5,018**	0.73	5,018	**5,018**	0.64	5,018	**5,018**	1.26	5,018
e2-b	6,317	**6,317**	0.94	6,318	**6,317**	0.79	6,321	**6,317**	2.25	6,317
e2-c	8,335	**8,335**	1.01	8,335	**8,335**	0.73	8,335	**8,335**	1.34	8,335
e3-a	5,898	**5,898**	0.98	5,898	**5,898**	0.95	5,898	**5,898**	1.99	5,898
e3-b	7,776	**7,775**	1.24	7,776	**7,775**	0.96	7,776	**7,775**	1.83	7,775
e3-c	10,293	**10,292**	1.39	10,292	**10,292**	0.95	10,292	**10,292**	2.25	10,292
e4-a	6,447	**6,444**	1.80	6,453	**6,444**	0.97	6,444	**6,444**	3.68	6,444
e4-b	8,994	8,976	2.46	8,986	**8,961**	1.45	8,985	**8,961**	3.10	8,961
e4-c	11,591	11,568	3.54	11,561	**11,529**	1.35	11,563	**11,529**	5.04	11,529
s1-a	5,018	**5,018**	0.73	5,018	**5,018**	0.80	5,018	**5,018**	1.38	5,018
s1-b	6,388	**6,388**	0.84	6,388	**6,388**	0.73	6,388	**6,388**	1.38	6,388
s1-c	8,518	**8,518**	0.75	8,518	**8,518**	0.67	8,518	**8,518**	1.20	8,518
s2-a	9,890	9,880	6.27	9,892	**9,868**	4.62	9,887	9,875	8.70	9,868
s2-b	13,124	13,067	12.13	13,105	13,081	6.21	13,102	13,081	7.59	13,057
s2-c	16,432	**16,425**	8.91	16,438	**16,425**	3.91	16,440	**16,425**	7.31	16,425
s3-a	10,220	**10,201**	7.38	10,233	**10,201**	3.83	10,240	10,221	7.92	10,201
s3-b	13,702	**13,682**	7.80	13,696	**13,682**	4.17	13,694	**13,682**	9.98	13,682
s3-c	17,219	**17,188**	13.65	17,234	17,189	5.11	17,191	**17,188**	8.93	17,188
s4-a	12,279	12,253	10.97	12,279	12,237	5.27	12,288	12,273	14.72	12,216
s4-b	16,326	16,257	16.08	16,287	16,253	7.11	16,284	16,230	18.18	16,213
s4-c	20,755	20,560	30.89	20,548	20,495	9.49	20,591	20,500	21.51	20,461
g1-a	995,798	992,995	22.26	996,135	**989,865**	16.04	993,127	992,227	34.80	989,865
g1-b	1,117,657	1,116,694	24.54	1,114,677	1,110,938	10.83	1,116,617	1,112,149	34.76	1,109,226
g1-c	1,241,740	1,235,831	40.68	1,236,375	1,231,384	16.24	1,236,062	1,232,501	44.39	1,230,155
g1-d	1,377,963	1,371,622	50.70	1,370,978	1,364,938	16.89	1,370,963	1,365,393	49.15	1,361,862
g1-e	1,523,226	1,515,288	53.53	1,512,422	1,504,992	15.84	1,511,572	1,503,467	49.52	1,501,801
g2-a	1,097,800	1,093,395	34.30	1,092,698	**1,086,899**	15.50	1,090,396	1,087,353	49.00	1,086,899
g2-b	1,207,950	1,201,163	33.19	1,203,715	1,198,118	15.62	1,202,901	1,198,633	48.76	1,196,873
g2-c	1,346,116	1,337,468	33.56	1,339,234	1,332,430	18.04	1,336,104	1,333,430	49.43	1,330,744
g2-d	1,485,101	1,479,785	42.13	1,473,417	1,469,759	15.54	1,476,285	1,471,783	50.36	1,468,118
g2-e	1,626,844	1,621,327	55.31	1,617,384	1,610,952	18.22	1,616,556	1,610,919	51.06	1,602,229
gap_{BKS}		0.24%			0.07%			0.10%		
T(min)			15.36			6.45			17.51	

Table 4: Results of the CARP on (24) EGL and (10) EGL-L instances, time in minutes.

For the extensions to the classical CARP instances, multiple depots are added following the proposed approach of Kansou and Yassine (2010). For the EGL and L-EGL instances, 4 depots are included at the nodes 1 and $x\lfloor\frac{|V|}{d-1}\rfloor$ for $x \in \{1, 2, 3\}$. Furthermore, a heterogeneous fleet is introduced for each instance. To this extent, a number of vehicle types $|T|$ is randomly drawn between 2 and 5. The heterogeneous fleet must adhere to two rules: (i) the heterogeneous fleet is composed such that the total number of available vehicles equals the original number of vehicles m for the instance and (ii) the total capacity of the constructed fleet should not exceed the total demand by more than twice the capacity of the smallest vehicle type (Taillard, 1999). The new heterogeneous instances are provided in the Appendix in Table 13.

The results for the extensions to the CARP are depicted in Tables 11 and 12 in the Appendix. These results were obtained by applying the T-GTBCX operator to the proposed MA framework. The average gap_{best} is depicted in the tables, which is equal to $100(z - z_{best})/z_{best}$, where z is the average solution value and z_{best} denotes the best solution value obtained by the 10 runs of the algorithm. While no references exist for the obtained results for the extended CARP, the results can be referenced by comparing the gap_{best} of the extended CARP to the CARP. The gap_{best} results for the CARP, OECARP and MDOECARP, depicted in Table 11, are 0.21%, 0.23% and 0.30% respectively. Thus, the OECARP and MDOECARP results, while producing a worse gap_{best}, seem to perform relatively similar in terms of their gap_{best}. Thus providing some confidence in the obtained results. The heterogeneous fleet extensions are depicted in Table 12. The MA produced a gap_{best} of 0.31%, 0.40% and 0.65% for the HCARP, HOECARP and HMDOECARP respectively. The gap_{best} of the heterogeneous fleet extensions worsened compared to the homogeneous fleet extensions, thus reducing the confidence in the solution qualities to some extent. Furthermore, in the heterogeneous CARP extensions, the running times significantly increased for the largest instances (ELG-L). This can be explained by the splitting procedure that is still present in the initialization of chromosomes, utilized by the diversification phase and the decomposition phase of the MA.

5.4 MO-CARP results

Finally, the performance of the implemented Non-dominated Sorting Algorithm (NSGA-II) framework for the MO-CARP is analyzed. To this extent, 2 performance measures are utilized. First, the inverted generational distance (I_D) proposed by Czyzżak and Jaszkiewicz (1998). The MO-CARP is solved by constructing a set of non-dominated solutions R. Then the I_D measures the average distance of R to the Pareto-front A. However, as the true Pareto-front A is unknown for instances, the Pareto-front for each instance is approximated by forming a set of non-dominated solutions from all obtained solutions over all considered algorithms. The inverted generational distance (I_D) is defined as

$$I_D(A) = \frac{\sum_{y \in R} \min_{x \in A} d(x, y)}{|R|}. \tag{27}$$

Where $d(x, y)$ denotes the euclidean distance between two solutions. Second, the hypervolume (I_H) performance measure depicts the area dominated by the solutions in a set of non-dominated solutions R (Zitzler et al., 2007). The hypervolume performance measure is one of most applied performance measures for multi-objective problems. As if a set dominates another set, the hypervolume of the dominates set is always lower (Brandão and Eglese, 2008). The I_H can be expressed as

$$I_H(R) = volume(\cup_{i=1}^{n}[v_i, r]). \tag{28}$$

Where n is the number of non-dominated solutions in the set R and $v_i \in \mathbb{N}^2$. Furthermore, $r \in \mathbb{N}^2$ denotes the reference point, the worst point such that $r_i = \max_{x \in \Omega} f_i(x)$, $\forall i = \{1, 2\}$ where Ω denotes the set of all obtained solutions over all algorithms. Figure 9 denotes the hypervolume for 3 solutions a, b and c in set R and a reference point r.

The two strategies tackling duplicate solution values, as discussed in Section 4, are tested for the developed framework to solve the MO-CARP. The first duplicate strategy D_{value} denotes the strategy where solutions of identical objective values are not allowed. While in the strategy $D_{distance}$, solutions of identical objective values are allowed. These strategies are compared to one another via the performance measures. Moreover, the results are further compared to the results of the DMEANS-2 algorithm (Zhang et al., 2020), a current state-of-the-art algorithm for MO-CARP. However, within the current body of literature

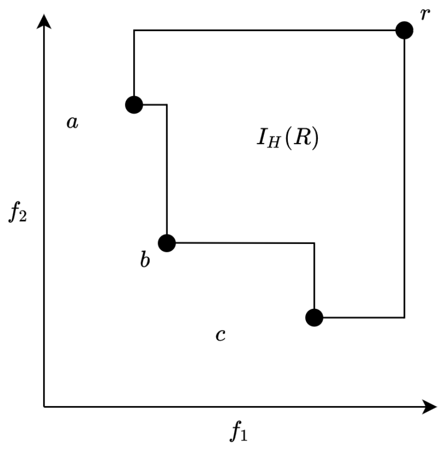


Figure 9: Hypervolume of non-dominated soltuin set R.

on the MO-CARP, the fleet size is relaxed. Thus, the results obtained by the NSGA-II for the I_D and I_H cannot be directly compared to the results of the DMEANS-2, as the Pareto-fronts can differ between the relaxed and fixed fleets. But, if the developed NSGA-II obtains solutions that dominate solutions of the DMEANS-2, then these solutions indicate a better performance. As the DMEANS-2 imposes no restrictions on the fleet size, where the developed NSGA-II does. Vice-versa, if a solution within the DMEANS-2 front dominates a solution of the developed NSGA-II front, this does not necessarily indicate a better performing algorithm. As potentially these solutions are unobtainable by imposing a fixed fleet.

The performance of the algorithms is evaluated on the EGL and the EGL-L instance sets using 30 repeats. The results for the developed NSGA-II algorithm are obtained under a time limit of 60 minutes, and the algorithm terminates whenever no improvement was identified in the first front in $it_{gen} = 30$ generations. Furthermore, the T-GTBCX operator is applied in the NSGA-II, the subpopulation size $\mu = 60$, neighborhood size of each arc $|\Gamma| = 20$, neighborhood size of routes for the SWAP* $|\Lambda| = 3$ and feasibility ratio $\xi^W = 0.2$. Each 10 generations, the front improvement strategy was executed. A diversification phase is initiated when the MA could not identify a improvement in the first front each $it_{div} = 12$ generations. Finally, the decomposition phase is performed each $it_{dec} = 6$ generations, counted from the last generation where an improvement was determined.

Table 5 depicts the results of the developed NSGA-II results on the EGL set. Furthermore, Table 6 depict the results for the EGL-L instance set. The DMEANS-2 algorithm, studied in Zhang et al. (2020), could only be used for the comparison on the EGL set, as it was not applied to the EGL-L set. The performance measure results are analyzed by the Wilcoxon signed-rank test, where significant better performance measure values between the D_{value} and $D_{distance}$ strategies are marked in bold. Furthermore, if a significant better performance measure value was obtained by a duplicate strategy compared to the DMEANS-2 algorithm, the values is underlined in the table. Figure 10 depicts the Pareto-fronts obtained over all obtained solutions sets for each algorithm on three instances. All obtained Pareto-fronts are included in the Appendix in Figures 11, 12 and 13.

		I_H			I_D			T(min)	
Instance	$\lvert R\rvert$	D_{value}	$D_{distance}$	DMEANS2	D_{value}	$D_{distance}$	DMEANS2	D_{value}	$D_{distance}$
e1-A	51	$\underline{0.708}$	$\underline{\mathbf{0.714}}$	0.647	0.006	0.005	0.007	0.59	0.81
e1-B	51	$\underline{0.372}$	$\underline{0.376}$	0.338	$\underline{\mathbf{0.003}}$	$\underline{0.004}$	0.008	0.44	0.70
e1-C	51	0.046	**0.093**	0.550	0.013	0.015	0.055	0.43	0.72
e2-A	72	0.609	**0.678**	0.779	0.008	0.003	0.005	1.09	1.66
e2-B	72	$\underline{0.791}$	$\underline{0.804}$	0.775	0.025	0.026	0.031	2.08	2.99
e2-C	72	0.360	0.364	0.547	$\underline{0.049}$	$\underline{0.054}$	0.077	1.54	2.40
e3-A	87	0.623	0.614	0.794	$\underline{0.011}$	$\underline{0.014}$	0.036	2.19	2.80
e3-B	87	$\underline{\mathbf{0.660}}$	$\underline{0.576}$	0.468	$\underline{0.027}$	$\underline{0.024}$	0.043	3.39	4.51
e3-C	87	$\underline{\mathbf{0.925}}$	$\underline{0.908}$	0.802	$\underline{0.045}$	$\underline{0.052}$	0.164	2.24	2.71
e4-A	98	$\underline{0.642}$	$\underline{\mathbf{0.685}}$	0.758	$\underline{0.013}$	$\underline{0.013}$	0.039	2.60	3.08
e4-B	98	$\underline{\mathbf{0.888}}$	$\underline{0.862}$	0.663	$\underline{0.026}$	$\underline{0.033}$	0.068	3.94	5.60
e4-C	98	$\underline{\mathbf{0.789}}$	$\underline{0.520}$	0.415	$\underline{\mathbf{0.084}}$	$\underline{0.232}$	0.209	3.83	4.68
s1-A	75	$\underline{0.314}$	$\underline{\mathbf{0.363}}$	0.860	$\underline{\mathbf{0.001}}$	$\underline{0.006}$	0.012	0.91	0.97
s1-B	75	$\underline{0.626}$	$\underline{\mathbf{0.643}}$	0.771	$\underline{\mathbf{0.011}}$	$\underline{0.021}$	0.029	2.22	3.09
s1-C	75	$\underline{0.834}$	$\underline{\mathbf{0.852}}$	0.812	$\underline{0.008}$	$\underline{0.007}$	0.019	1.63	1.77
s2-A	147	$\underline{0.603}$	$\underline{\mathbf{0.642}}$	0.744	$\underline{0.012}$	$\underline{0.013}$	0.035	23.14	12.57
s2-B	147	$\underline{\mathbf{0.750}}$	$\underline{0.719}$	0.719	$\underline{0.044}$	$\underline{0.047}$	0.074	32.33	19.08
s2-C	147	$\underline{0.628}$	$\underline{0.622}$	0.396	$\underline{0.067}$	$\underline{0.062}$	0.199	23.29	11.73
s3-A	159	$\underline{0.505}$	$\underline{\mathbf{0.669}}$	0.758	$\underline{0.018}$	$\underline{0.016}$	0.043	17.52	12.28
s3-B	159	$\underline{0.767}$	$\underline{\mathbf{0.788}}$	0.683	$\underline{\mathbf{0.024}}$	$\underline{0.033}$	0.097	29.87	16.30
s3-C	159	$\underline{0.653}$	$\underline{0.671}$	0.581	0.181	0.151	0.188	27.05	16.58
s4-A	190	0.447	0.449	0.455	$\underline{0.101}$	$\underline{0.102}$	0.171	19.14	14.75
s4-B	190	0.454	0.452	0.425	$\underline{0.590}$	$\underline{0.548}$	0.845	17.00	8.22
s4-C	190	$\underline{0.525}$	$\underline{0.626}$	0.426	0.371	0.319	0.286	34.38	18.00
Average	-	0.605	0.612	0.632	0.072	0.075	0.114	10.54	7.00

Table 5: Average results of the MO-CARP on EGL instances, time in minutes.

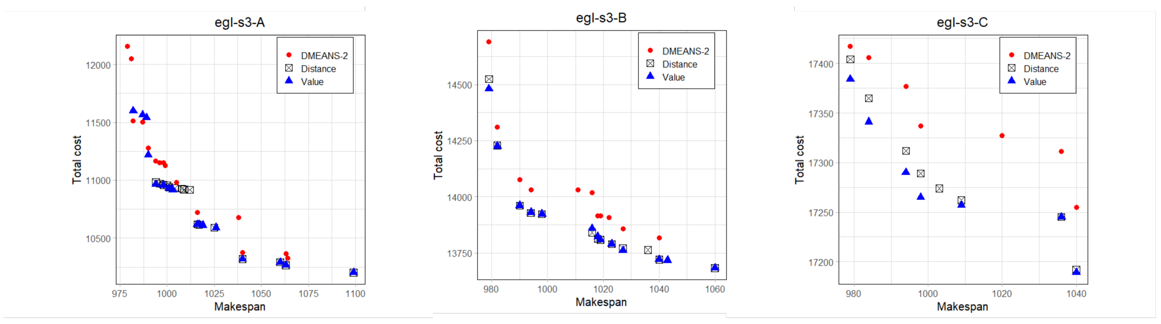


Figure 10: Pareto-front for the NSGA-II with various duplicate management strategies and DMEANS-2.

From the EGL instance results, the developed NSGA-II in this thesis performs relatively better compared to the DMEANS-2. First, on 18 of the 24 EGL instances, statistical improvements were obtained in terms hypervolume I_H (underlined in the table). Second, the NSGA-II improved the I_D statistically significant on 18 of the 24 EGL instances, indicating that results obtained by the NSGA-II construct solution sets that better match the the Pareto-front compared to the DMEANS-2. These results can be visually inspected in Figures 11 and 12 in the Appendix. Furthermore, the NSGA-II algorithm produces better results in total cost for most instances, while the NSGA-II is in some instances dominated in terms of makespan. However, this was to be expected, as the DMEANS-2 concerned itself with a relaxed fleet, and thus is able to balance the routes over more vehicles. The authors

		I_H		I_D		T(min)	
Instance	$\|R\|$	D_{value}	$D_{distance}$	D_{value}	$D_{distance}$	D_{value}	$D_{distance}$
g1-A	347	0.730	0.714	0.066	0.076	45.91	48.86
g1-B	347	**0.871**	0.827	0.040	0.049	34.09	31.83
g1-C	347	0.848	0.843	0.068	0.084	38.97	41.84
g1-D	347	0.742	**0.788**	0.113	0.094	54.21	57.41
g1-E	347	0.743	0.768	0.094	0.106	52.65	46.42
g2-A	375	**0.672**	0.572	0.134	0.140	41.02	46.23
g2-B	375	**0.832**	0.726	0.197	0.160	36.44	44.14
g2-C	375	**0.815**	0.810	0.133	0.161	50.41	49.72
g2-D	375	0.705	0.674	0.210	0.200	45.93	53.47
g2-E	375	0.547	0.558	0.320	0.470	52.73	50.32
Average	-	0.750	0.728	0.137	0.154	45.24	47.02

Table 6: Results of the MO-CARP on EGL-L instances, time in minutes.

of the DMEANS-2 did not provide details on the computational time, thus no comparison in terms of computational time can be made.

The results on the EGL instances show that the duplicate management strategy $D_{distance}$ performed relatively well on the larger instances ($|R| > 100$), while no decisive strategy can be identified in the smaller instances. The D_{value} strategy provided better results in terms of the I_D, as 5 significant improvements over the $D_{distance}$ were identified, while the $D_{distance}$ did not obtain better values of I_D in any instance. Finally, the computational times show that the $D_{distance}$ strategy runs faster compared than the D_{value} on larger instances, while on smaller instances the D_{value} strategy terminates quicker.

The results of the duplicate strategies on the EGL-L instances show that the D_{value} strategy obtained 4 statistically significant better values of I_H over 10 instances. While $D_{distance}$ obtained only 1 significant improvement in terms of I_H. Furthermore, no significant differences were observed in terms of the I_D between the two strategies. Finally, the D_{value} terminated, slightly, faster than the $D_{distance}$ on average, reversing the results of the running times observed in the EGL instances.

The results of the MO-CARP instances indicate that the developed NSGA-II framework performs well compared to the DMEANS-2, as new solutions for the Pareto-front were identified on most instances. This result is shown by the observed significant improvements in the performance measures I_H and I_D. Furthermore, the duplicate strategy $D_{distance}$ obtains better results in terms of I_H compared to the D_{value} strategy on the larger instances of the EGL set, but these results did not transfer to the large instance set EGL-L. A possible explanation for this effect is the convergence of the solutions within the $D_{distance}$ strategy. The solution set within $D_{distance}$ is able to cluster on all non-dominated solutions in the first front, while D_{value} maintains a more diverse solution set and thus is possibly able to explore a more diverse solution space for the larger instances.

6 Conclusion

In this thesis, various contributions to the Arc Routing Problem are presented. First, the CARP is extended to a variation of the OCARP, where routes must start from a depot but need not necessarily end at a depot, the Open-End CARP (OECARP). Hereafter, the OECARP is extended by multiple-depots (MDOECARP) and hereafter the OECARP and

MDOECARP are considered with a heterogeneous fleet of vehicles. Finally, the CARP is extended to account for multiple-objectives while still imposing a fixed fleet. The OCARP has been studied in the body of literature on ARPs, while the OECARP and its extensions proposed in this thesis have not yet been examined.

The CARP and its extensions are solved using state-of-the-art local search heuristics and Evolutionary Algorithm population management techniques from the VRP literature. Within this thesis, a new splitting procedure for heterogeneous fleets is proposed. The experiments showed a speed-up compared to the classical splitting algorithms. Larger speed-ups are observed for larger instances and fleet compositions with small differences in capacities between vehicle types. Hereafter, a novel crossover operator, the Tabu Giant Tour Best Cost Crossover (T-GTBCX) is proposed. Furthermore, within the ARP literature, no study was performed on the effectiveness of crossover operators within an Evolutionary Algortihm. In this thesis, this analysis is performed for classical and the proposed crossovers. The results showed that effective crossovers within the VRP literature did not transfer directly to ARPs. The novel T-GTBCX was found to be a very effective crossover, as it dominated all other tested crossovers in solution value and running time.

The T-GTBCX was able to improve 4 BKS of the largest benchmark instance sets of the CARP literature. Furthermore, the proposed MA framework paired with the T-GTBCX provided, on average, better results than the current state-of-the-art techniques for the CARP. The extension of the CARP to a heterogeneous fleet, open-end and multi-depots showed a slight decline in solution value quality, indicated by an increase in the gap between the average and best solution over 10 runs for each tested instance. However, this is to be expected, as the solution space increases due to the extensions.

Hereafter, the CARP is extended to account for multiple-objectives: the total cost and makespan. A NSGA-II based EA was enhanced by recent developments within the EAs for routing problems. A new strategy to handle duplicate solution values are proposed within this thesis. Where a set of duplicate solutions is maintained for each unique solution value. Within the set, diversity is ensured by the broken pairs distance metric. The NSGA-II was shown to outperform a recently proposed DMEANS-2 for the MOCARP in most instances. Furthermore, the classical and new proposed duplicate management strategies performed well on the test instances. However the duplicate management strategy where duplicates were discarded, performed relatively better on the largest instances based on performance metrics.

The obtained results within this thesis somewhat contradict each other, as a novel heterogeneous splitting procedure was proposed, while a crossover was proposed that did not require a splitting procedure. However, the proposed T-GTBCX operator produced solutions of such a quality in terms of solution value and running time, that future research could focus on investigating the T-GTBCX operator in more detail. Furthermore, most proposed extensions to the CARP within this thesis cannot be referenced to existing algorithms. Thus, an interesting direction of research would be the implementation and adapt various state-of-the-art algorithms to compare the results. Moreover, the discussed results were obtained on instances from the ARP literature, which are relatively small compared to problems encountered in real-world ARPs. Thus, future research could investigate the effectiveness of the proposed contributions on (large-scale) real-world instances.

Finally, the developed techniques seems to be well suited for the preventive gritting problem of bridges that the municipality of Rotterdam currently faces. The municipality requires service of a total of 362 streets for their bridge routing problem. Thus this instance size is similar to the tested EGL-L instances within this thesis. Furthermore. the minimum number of vehicles can be asserted by solving the (HMD)OECARP while iteratively reducing the fleet size. The trade-off between total cost and makespan can hereafter be visualized for the decision makers by applying the proposed NSGA-II MO-CARP framework on the desired fleet.

References

Abdoun, O. and Abouchabaka, J. (2012). A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *CoRR*, abs/1203.3097.

Amberg, A., Domschke, W., and Voß, S. (2000). Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *Eur. J. Oper. Res.*, 124(2):360–376.

Arakaki, R. K. and Usberti, F. L. (2019). An efficiency-based path-scanning heuristic for the capacitated arc routing problem. *Computers Operations Research*, 103:288–295.

Arnold, F., Santana, Í., Sörensen, K., and Vidal, T. (2021). Pils: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recognit.*, 116:107957.

Beasley, J. (1983). Route first–Cluster second methods for vehicle routing. *Omega*, 11(4):403–408.

Belenguer, J. M. (1990). The Capacitated arc routing problem polyhedron. *Ph.D. dissertation.*

Belenguer, J. M. and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers Operations Research*, 30:705–728.

Belenguer, J. M., Benavent, E., and Irnich, S. (2015). Chapter 9: The capacitated arc routing problem: Exact algorithms. In *Arc Routing*, pages 183–221. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Beullens, P., Muyldermans, L., and Cattrysse, D. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147:629–643.

Borwein, J. M. (1983). On the existence of pareto efficient points. *Mathematics of Operations Research*, 8(1):64–73.

Brandão, J. and Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem (carp). *Computers Operations Research*, 35:1112–1126.

Chu, F., Labadie, N., and Prins, C. (2006). A scatter search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169:586–605.

Corberán, A., Marti, R., and Romero, A. (2000). Heuristics for the mixed rural postman problem. *Computers OR*, 27:183–203.

Czyzżak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Desaulniers, G., Desrosiers, J., and Solomon, M. (2005). *Column Generation.*

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Oper. Res.*, 42:977–978.

Duhamel, C., Lacomme, P., Prins, C., and Prodhon, C. (2010). A grasp×els approach for the capacitated location-routing problem. *Comput. Oper. Res.*, 37(11):1912–1923.

Duhamel, C., Lacomme, P., and Prodhon, C. (2011). Efficient frameworks for greedy split and new depth first search split procedures for routing problems. *Comput. Oper. Res.*, 38(4):723–739.

Duhamel, C., Lacomme, P., and Prodhon, C. (2012). A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. *Eng. Appl. Artif. Intell.*, 25(2):345–358.

Dupin, N. and Talbi, E. (2018). Clustering in a 2-dimensional pareto front: the p-median and p-center problems are polynomially solvable. *CoRR*, abs/1806.02098.

Eglese, R. (1994). Routeing winter gritting vehicles. *Discrete Applied Mathematics*, 48(3):231–244.

Eglese, R. W. and Li, L. Y. O. (1996). *A Tabu Search based Heuristic for Arc Routing with a Capacity Constraint and Time Deadline*, pages 633–649. Springer US, Boston, MA.

Fleury, G., Lacomme, P., and Prins, C. (2004). Evolutionary algorithms for stochastic arc routing problems. pages 501–512.

Ghiani, G., Guerriero, F., Improta, G., and Musmanno, R. (2005). Waste collection in southern italy: solution of a real-life arc routing problem. *Int. Trans. Oper. Res.*, 12(2):135–144.

Ghiani, G., Improta, G., and Laporte, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37:134–143.

Gillett, B. E. and Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, 22(2):340–349.

Golden, B., Dearmon, J., and Baker, E. (1983). Computational experiments with algorithms for a class of routing problems. *Computers Operations Research*, 10(1):47–59.

Golden, B., Raghavan, S., and Wasil, E., editors (2008). *The vehicle routing problem: Latest advances and new challenges*. Springer US, Boston, MA.

Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

Grandinetti, L., Guerriero, F., Laganà, D., and Pisacane, O. (2012). An optimization-based heuristic for the multi-objective undirected capacitated arc routing problem. *Computers OR*, 39:2300–2309.

Gómez-Cabrero, D., Belenguer, J., and Benavent, E. (2005). Cutting plane and column generation for the capacitated arc routing problem. *Presented at ORP3 ,Valencia.*

Helsgaun, K. (2017). An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University.

Hertz, A., Laporte, G., and Mittaz, M. (1999). A tabu search heuristic for the capacitated arc routing problem. *Operations research*, 48(1):129–135.

Hertz, A. and Mittaz, M. (2001). A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35:425–434.

Kansou, A. and Yassine, A. (2009). A two ant colony approaches for the multi-depot capacitated arc routing problem. In *2009 International Conference on Computers & Industrial Engineering*. IEEE.

Kansou, A. and Yassine, A. (2010). New upper bounds for the multi-depot capacitated arc routing problem. *International Journal of Metaheuristics*, 1.

Lacomme, P., Prins, C., and Ramdane Cherif-Khettaf, W. (2001). A genetic algorithm for the capacitated arc routing problem and its extensions. volume 2037, pages 473–483.

Lacomme, P., Prins, C., and Ramdane Cherif-Khettaf, W. (2004a). Competitive memetic algorithms for arc routing problems. *Annals OR*, 131:159–185.

Lacomme, P., Prins, C., and Ramdane Cherif-Khettaf, W. (2005). Evolutionary algorithms for periodic arc routing problems. *European Journal of Operational Research*, 165:535–553.

Lacomme, P., Prins, C., and Sevaux, M. (2006). A genetic algorithm for a bi-objective capacitated arc routing problem. *Comput. Oper. Res.*, 33(12):3473–3493.

Lacomme, P., Prins, C., and Tanguy, A. (2004b). First competitive ant colony scheme for the carp. pages 426–427.

Laporte, G., Røpke, S., and Vidal, T. (2014). *Heuristics for the Vehicle Routing Problem*, pages 87–116. M O S - S I A M Series on Optimization. Society for Industrial and Applied Mathematics, 2 edition.

Liu, R., Jiang, Z., Geng, N., and Liu, T. (2010). A heuristic approach for the open capacitated arc routing problem.

Liu, T., Jiang, Z., and Geng, N. (2014). A genetic local search algorithm for the multi-depot heterogeneous fleet capacitated arc routing problem. *Flex. Serv. Manuf. J.*, 26(4):540–564.

Longo, H., de Aragão, M. P., and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the cvrp. *Comput. Oper. Res.*, 33(6):1823–1837.

Luiz Usberti, F., Morelato França, P., and França, A. L. M. (2013). Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers Operations Research*, 40(12):3206–3217.

Mei, Y., Tang, K., and Yao, X. (2011). Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 15(2):151–165.

Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341.

Pearn, W. L. (1991). Augment-insert algorithms for the capacitated arc routing problem. *Computers & Operations Research*, 18(2):189–198.

Pia, A. and Filippi, C. (2006). A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. *Int. Trans. Oper. Res.*, 13(2):125–141.

Polacek, M., Benkner, S., Doerner, K., and Hartl, R. (2008). A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. *BuR : Business Research*, 1.

Prins, C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6):916–928. Artificial Intelligence Techniques for Supply Chain Management.

Prins, C., Lacomme, P., and Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transp. Res. Part C Emerg. Technol.*, 40:179–200.

Puljić, K. and Manger, R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18.

Ramdane-Cherif, W. (2006). Evolutionary algorithms for capacitated arc routing problems with time windows. *IFAC proc. vol.*, 39(3):321–326.

Sajid, M., Singh, J., Haidri, R. A., Prasad, M., Varadarajan, V., Kotecha, K., and Garg, D. (2021). A novel algorithm for capacitated vehicle routing problem for smart cities. *Symmetry*, 13(10).

Santini, A., Schneider, M., Vidal, T., and Vigo, D. (2021). Decomposition strategies for vehicle routing heuristics.

Santos, L., Coutinho-Rodrigues, J., and Current, J. R. (2009). An improved heuristic for the capacitated arc routing problem. *Comput. Oper. Res.*, 36(9):2632–2637.

Taillard, E. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO - Operations Research - Recherche Opérationnelle*, 33(1):1–14.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15:333–346.

Ulusoy, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337.

Usberti, F. L., França, P. M., and França, A. L. M. (2011). The open capacitated arc routing problem. *Computers Operations Research*, 38(11):1543–1555.

Vidal, T. (2015). Technical note: Split algorithm in o(n) for the vehicle routing problem. *CoRR*, abs/1508.02759.

Vidal, T. (2017). Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension. *Oper. Res.*, 65(4):992–1010.

Vidal, T. (2022). Hybrid genetic search for the capacitated vehicle routing. *Comput. Oper. Res.*, 140:105643.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234:658–673.

Welz, S. A. (1994). *Optimal solutions for the capacitated arc routing problem using integer programming.* PhD thesis.

Wøhlk, S. (2005). Contributions to arc routing.

Wøhlk, S. (2008). A Decade of Capacitated Arc Routing. *Operations Research/Computer Science Interfaces*, pages 29–48.

Xing, L., Rohlfshagen, P., Chen, Y., and Yao, X. (2010). An evolutionary approach to the multidepot capacitated arc routing problem. *IEEE Trans. Evol. Comput.*, 14(3):356–374.

Zhang, Q., Wu, F., Tao, Y., Pei, J., Liu, J., and Yao, X. (2020). D-maens2: A self-adaptive d-maens algorithm with better decision diversity. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2754–2761.

Zitzler, E., Brockhoff, D., and Thiele, L. (2007). The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Lecture Notes in Computer Science*, pages 862–876. Springer Berlin Heidelberg.

Appendix

	Crossover operator							
Instance	OX	HGreX	CX	AEX	PMX	MIX	GTBCX	T-GTBCX
egl-e4-C	11742.1	11870.6	11613.1	11957.9	11617.6	11716.9	12092.2	11592.8
	1.85%	2.96%	0.73%	3.72%	0.77%	1.63%	4.89%	0.55%
egl-s2-A	10075.0	10121.0	9970.5	10171.5	9991.4	10047.3	10164.3	9899.1
	2.10%	2.56%	1.04%	3.08%	1.25%	1.82%	3.00%	0.32%
egl-s3-A	10350.8	10417.2	10294.8	10543.3	10310.1	10381.5	10505.8	10250.8
	1.47%	2.12%	0.92%	3.36%	1.07%	1.77%	2.99%	0.49%
egl-s4-A	12528.0	12624.3	12433.1	12743.8	12462.4	12533.5	12784.9	12298.6
	2.55%	3.34%	1.78%	4.32%	2.02%	2.60%	4.66%	0.68%
average	1.99%	2.75%	1.12%	3.62%	1.28%	1.95%	3.88%	0.51%

Table 7: Crossover comparison with 1% mutation.

	Crossover operator							
Instance	OX	HGreX	CX	AEX	PMX	MIX	GTBCX	T-GTBCX
egl-e4-C	11761.3	11872.9	11628.2	11946.8	11624.9	11748.3	12096.4	11594.7
	2.01%	2.98%	0.86%	3.62%	0.83%	1.90%	4.92%	0.57%
egl-s2-A	10079.1	10118.9	9988.0	10185.3	9980.0	10068.7	10139.3	9900.2
	2.14%	2.54%	1.22%	3.22%	1.13%	2.03%	2.75%	0.33%
egl-s3-A	10369.5	10411.9	10316.8	10529.3	10295.5	10362.8	10499.6	10257.0
	1.65%	2.07%	1.13%	3.22%	0.93%	1.59%	2.93%	0.55%
egl-s4-A	12536.5	12629.2	12455.7	12769.6	12457.7	12530.0	12845.5	**12296.0**
	2.62%	3.38%	1.96%	4.53%	1.98%	2.57%	5.15%	0.65%
average	2.11%	2.74%	1.29%	3.65%	1.22%	2.02%	3.94%	0.52%

Table 8: Crossover comparison with 5% mutation.

	Crossover operator							
	OX	HGreX	CX	AEX	PMX	MIX	GTBCX	T-GTBCX
egl-e4-C	11748.0	11883.7	11640.3	11959.6	11657.5	11749.4	12135.4	11594.6
	1.90%	3.08%	0.97%	3.74%	1.11%	1.91%	5.26%	0.57%
egl-s2-A	10080.6	10132.4	9996.2	10183.6	9997.7	10050.0	10150.0	9894.8
	2.15%	2.68%	1.30%	3.20%	1.31%	1.84%	2.86%	0.27%
egl-s3-A	10363.2	10405.8	10307.3	10522.8	10307.4	10373.4	10501.5	10259.2
	1.59%	2.01%	1.04%	3.15%	1.04%	1.69%	2.95%	0.57%
egl-s4-A	12550.6	12639.4	12449.7	12762.0	12479.0	12560.5	12810.7	12308.4
	2.74%	3.47%	1.91%	4.47%	2.15%	2.82%	4.87%	0.76%
average	2.10%	2.81%	1.30%	3.64%	1.41%	2.07%	3.98%	0.54%

Table 9: Crossover comparison with 10% mutation.

	Crossover operator							
	OX	HGreX	CX	AEX	PMX	MIX	GTBCX	T-GTBCX
egl-e4-C	0.63	0.12	0.15	0.90	0.08	0.95	0.20	0.23
egl-s2-A	0.90	0.45	0.23	0.48	0.61	0.51	0.38	0.34
egl-s3-A	0.25	0.50	**0.01**	0.21	0.23	0.25	0.35	0.63
egl-s4-A	0.27	0.64	0.12	0.27	0.09	0.46	0.07	**0.04**

Table 10: ANOVA test p-values for each crossover type over all 4 mutation probabilities for each instance, H_0 : no difference in means.

	CARP			OECARP			MDOECARP		
Instance	Average	Best	T(min)	Average	Best	T(min)	Average	Best	T(min)
egl-e1-a	3,548	3,548	0.47	2,304	2,304	0.49	1,993	1,993	0.66
egl-e1-b	4,498	4,498	0.44	2,630	2,630	0.43	2,093	2,093	0.67
egl-e1-c	5,595	5,595	0.50	3,170	3,170	0.41	2,213	2,211	0.62
egl-e2-a	5,018	5,018	0.64	3,018	3,018	0.78	2,564	2,562	1.25
egl-e2-b	6,318	6,317	0.79	3,661	3,659	0.75	2,620	2,620	0.81
egl-e2-c	8,335	8,335	0.73	4,551	4,550	0.63	3,029	3,029	0.87
egl-e3-a	5,898	5,898	0.95	3,633	3,633	0.78	2,779	2,772	1.31
egl-e3-b	7,776	7,775	0.96	4,468	4,461	0.94	2,909	2,909	1.09
egl-e3-c	10,292	10,292	0.95	5,574	5,567	0.86	3,391	3,390	1.09
egl-e4-a	6,453	6,444	0.97	3,952	3,952	0.93	2,920	2,920	1.30
egl-e4-b	8,986	8,961	1.45	5,106	5,092	1.12	3,220	3,212	1.14
egl-e4-c	11,561	11,529	1.35	6,258	6,244	1.15	3,821	3,802	1.67
egl-s1-a	5,018	5,018	0.80	2,999	2,999	0.77	2,205	2,205	0.83
egl-s1-b	6,388	6,388	0.73	3,549	3,549	0.69	2,351	2,351	0.92
egl-s1-c	8,518	8,518	0.67	4,557	4,557	0.93	2,759	2,759	0.88
egl-s2-a	9,892	9,868	4.62	6,232	6,222	2.87	4,506	4,491	3.47
egl-s2-b	13,105	13,081	6.21	7,612	7,593	3.89	5,008	4,992	4.14
egl-s2-c	16,438	16,425	3.91	9,230	9,215	3.72	5,638	5,622	4.56
egl-s3-a	10,233	10,201	3.83	6,408	6,391	3.83	4,611	4,610	4.17
egl-s3-b	13,696	13,682	4.17	7,991	7,977	4.93	5,178	5,167	4.46
egl-s3-c	17,234	17,189	5.11	9,686	9,662	6.78	5,901	5,885	4.99
egl-s4-a	12,279	12,237	5.27	7,553	7,519	4.35	5,296	5,289	4.27
egl-s4-b	16,287	16,253	7.11	9,344	9,304	5.36	5,957	5,936	4.63
egl-s4-c	20,548	20,495	9.49	11,519	11,467	8.44	7,141	7,075	10.06
egl-g1-a	996,135	989,865	16.04	735,547	733,182	12.27	660,636	656,216	13.17
egl-g1-b	1,114,677	1,110,938	10.83	772,315	770,775	11.26	680,170	676,106	10.83
egl-g1-c	1,236,375	1,231,384	16.24	822,197	816,123	12.26	711,244	708,089	14.15
egl-g1-d	1,370,978	1,364,938	16.89	880,674	876,688	10.44	748,353	744,088	16.01
egl-g1-e	1,512,422	1,504,992	15.84	945,342	941,335	12.36	793,372	789,036	17.29
egl-g2-a	1,092,698	1,086,899	15.50	793,983	790,420	14.69	717,734	712,253	22.87
egl-g2-b	1,203,715	1,198,118	15.62	832,964	828,275	16.42	735,632	729,826	18.05
egl-g2-c	1,339,234	1,332,430	18.04	888,252	882,870	17.05	770,793	766,598	22.15
egl-g2-d	1,473,417	1,469,759	15.54	946,913	945,167	20.30	810,834	804,342	22.94
egl-g2-e	1,617,384	1,610,952	18.22	1,007,291	1,003,048	22.58	851,730	847,850	19.79
gap_{best}		0.21%			0.23%			0.30%	

Table 11: Results of the homogeneous fleet CARP extensions on EGL and EGL-L instance, time in minutes.

	HCARP			HOECARP			HMDOECARP		
Instance	Average	Best	T(min)	Average	Best	T(min)	Average	Best	T(min)
egl-e1-a	3,415	3,415	0.54	2,274	2,274	0.75	1,967	1,967	1.16
egl-e1-b	4,305	4,305	0.54	2,562	2,562	0.62	2,016	2,016	0.92
egl-e1-c	5,883	5,880	0.78	3,278	3,277	0.71	2,355	2,349	0.88
egl-e2-a	4,822	4,822	0.68	3,018	3,018	0.81	2,462	2,456	1.29
egl-e2-b	5,972	5,969	0.86	3,440	3,440	0.92	2,550	2,550	1.21
egl-e2-c	7,954	7,948	0.91	4,374	4,360	1.01	2,875	2,868	1.29
egl-e3-a	5,694	5,693	1.28	3,525	3,525	0.94	2,671	2,668	1.53
egl-e3-b	7,193	7,190	1.09	4,151	4,150	1.00	2,765	2,765	1.19
egl-e3-c	10,391	10,371	1.57	5,698	5,675	1.28	3,613	3,575	1.98
egl-e4-a	6,147	6,142	1.59	3,843	3,843	1.15	2,849	2,845	1.50
egl-e4-b	8,192	8,183	1.41	4,704	4,694	1.69	3,040	3,024	1.51
egl-e4-c	11,424	11,385	1.95	6,262	6,229	1.87	3,841	3,759	3.20
egl-s1-a	4,918	4,918	0.78	2,880	2,880	0.87	2,141	2,138	1.01
egl-s1-b	6,632	6,620	1.23	3,678	3,657	1.21	2,554	2,543	1.45
egl-s1-c	7,963	7,960	1.25	4,269	4,264	1.06	2,614	2,609	1.21
egl-s2-a	9,388	9,382	4.73	5,977	5,970	4.00	4,410	4,401	4.72
egl-s2-b	12,549	12,475	6.16	7,402	7,347	7.12	4,938	4,895	7.35
egl-s2-c	15,228	15,179	9.03	8,617	8,581	6.39	5,379	5,342	7.32
egl-s3-a	10,340	10,315	4.75	6,506	6,453	6.31	4,669	4,639	5.24
egl-s3-b	13,443	13,359	12.76	7,890	7,819	10.63	5,249	5,184	11.25
egl-s3-c	15,952	15,892	9.48	9,079	9,017	6.48	5,594	5,556	8.82
egl-s4-a	12,021	11,979	8.32	7,484	7,417	13.01	5,394	5,325	12.34
egl-s4-b	15,730	15,643	8.90	9,173	9,118	9.06	6,008	5,954	9.35
egl-s4-c	18,742	18,628	14.75	10,559	10,534	17.10	6,587	6,533	17.46
egl-g1-a	992,690	988,169	12.22	742,724	735,912	16.13	672,676	667,294	22.70
egl-g1-b	1,097,222	1,092,582	14.64	767,269	763,841	19.41	679,779	671,495	23.26
egl-g1-c	1,197,834	1,189,361	38.01	809,337	805,417	48.05	702,432	697,598	51.01
egl-g1-d	1,316,429	1,308,044	41.94	860,425	854,479	53.41	726,092	721,110	60.00
egl-g1-e	1,432,728	1,422,548	60.00	914,206	910,608	60.00	765,142	755,690	60.00
egl-g2-a	1,070,610	1,062,461	14.99	783,205	778,589	15.77	708,755	701,378	25.01
egl-g2-b	1,196,436	1,190,278	16.54	832,749	827,021	23.37	737,214	730,243	28.63
egl-g2-c	1,294,406	1,286,919	31.80	877,244	872,231	31.74	759,395	754,159	38.12
egl-g2-d	1,414,674	1,410,146	49.73	925,792	920,500	54.89	788,316	782,620	59.23
egl-g2-e	1,542,546	1,532,947	60.00	989,858	983,015	60.00	834,537	825,957	60.00
gap_{best}		0.31%			0.40%			0.65%	

Table 12: Results of the heterogeneous fleet CARP extensions on EGL and EGL-L instance, time in minutes.

Instance	$\|V\|$	$\|R\|$	m	W	Old BKS	New BKS	$\vec{a}$	$\vec{c}$
E1-A	78	51	5	305	3548	3548	[3, 2]	[305, 346]
E1-B	78	51	7	220	4498	4498	[5, 2]	[220, 249]
E1-C	78	51	10	160	5595	5595	[1, 4, 5]	[122, 138, 160]
E2-A	78	72	7	280	5018	5018	[5, 2]	[280, 318]
E2-B	78	72	10	200	6317	6317	[7, 2, 1]	[200, 227, 263]
E2-C	78	72	14	140	8335	8335	[12, 1, 1]	[140, 159, 184]
E3-A	78	87	8	280	5898	5898	[5, 3]	[280, 318]
E3-B	78	87	12	190	7775	7775	[8, 2, 2]	[190, 215, 249]
E3-C	78	87	17	135	10292	10292	[1, 8, 8]	[107, 121, 140]
E4-A	78	98	9	280	6444	6444	[5, 4]	[280, 318]
E4-B	78	98	14	180	8961	8961	[9, 3, 2]	[180, 204, 236]
E4-C	78	98	19	130	11529	11529	[1, 3, 12, 3]	[99, 112, 130, 154]
S1-A	141	75	7	210	5018	5018	[5, 2]	[210, 238]
S1-B	141	75	10	150	6388	6388	[1, 3, 6]	[114, 129, 149]
S1-C	141	75	14	103	8518	8518	[10, 3, 1]	[103, 117, 135]
S2-A	141	147	14	235	9875	9868	[10, 3, 1]	[235, 266, 309]
S2-B	141	147	20	160	13057	13057	[3, 3, 8, 6]	[122, 138, 160, 189]
S2-C	141	147	27	120	16425	16425	[15, 2, 5, 5]	[105, 119, 138, 163]
S3-A	141	159	15	240	10201	10201	[3, 1, 11]	[183, 207, 240]
S3-B	141	159	22	160	13682	13682	[4, 7, 5, 6]	[122, 138, 160, 189]
S3-C	141	159	29	120	17188	17188	[20, 2, 4, 2, 1]	[109, 123, 142, 168, 203]
S4-A	141	190	19	230	12216	12216	[6, 2, 6, 5]	[175, 198, 230, 273]
S4-B	141	190	27	160	16214	16213	[6, 1, 15, 5]	[122, 138, 160, 189]
S4-C	141	190	35	120	20461	20461	[20, 6, 3, 4, 2]	[105, 119, 138, 163, 197]
G1-A	256	347	20	28600	991176	989865	[1, 6, 11, 2]	[21860, 24835, 28850, 34253]
G1-B	256	347	25	22800	1109656	1109226	[2, 9, 8, 6]	[17427, 19798, 22999, 27306]
G1-C	256	347	30	19000	1230155	1230155	[9, 7, 6, 4, 4]	[14522, 16498, 19165, 22754, 27598]
G1-D	256	347	35	16200	1361862	1361862	[12, 5, 5, 10, 3]	[12382, 14067, 16341, 19401, 23531]
G1-E	256	347	40	14100	1501801	1501801	[9, 13, 8, 2, 8]	[10777, 12243, 14222, 16885, 20479]
G2-A	256	375	22	28000	1086932	1086899	[6, 4, 3, 9]	[21401, 24313, 28244, 33534]
G2-B	256	375	27	23100	1196873	1196873	[1, 12, 8, 6]	[17656, 20058, 23301, 27665]
G2-C	256	375	32	19400	1330744	1330744	[5, 9, 13, 1, 4]	[14828, 16846, 19569, 23234, 28180]
G2-D	256	375	37	16700	1468310	1468118	[9, 4, 14, 8, 2]	[12764, 14501, 16845, 20000, 24258]
G2-E	256	375	42	14700	1602229	1602229	[16, 8, 3, 5, 10]	[10915, 12400, 14405, 17103, 20744]

Table 13: Characteristics of the (24) EGL and (10) EGL-L instances.

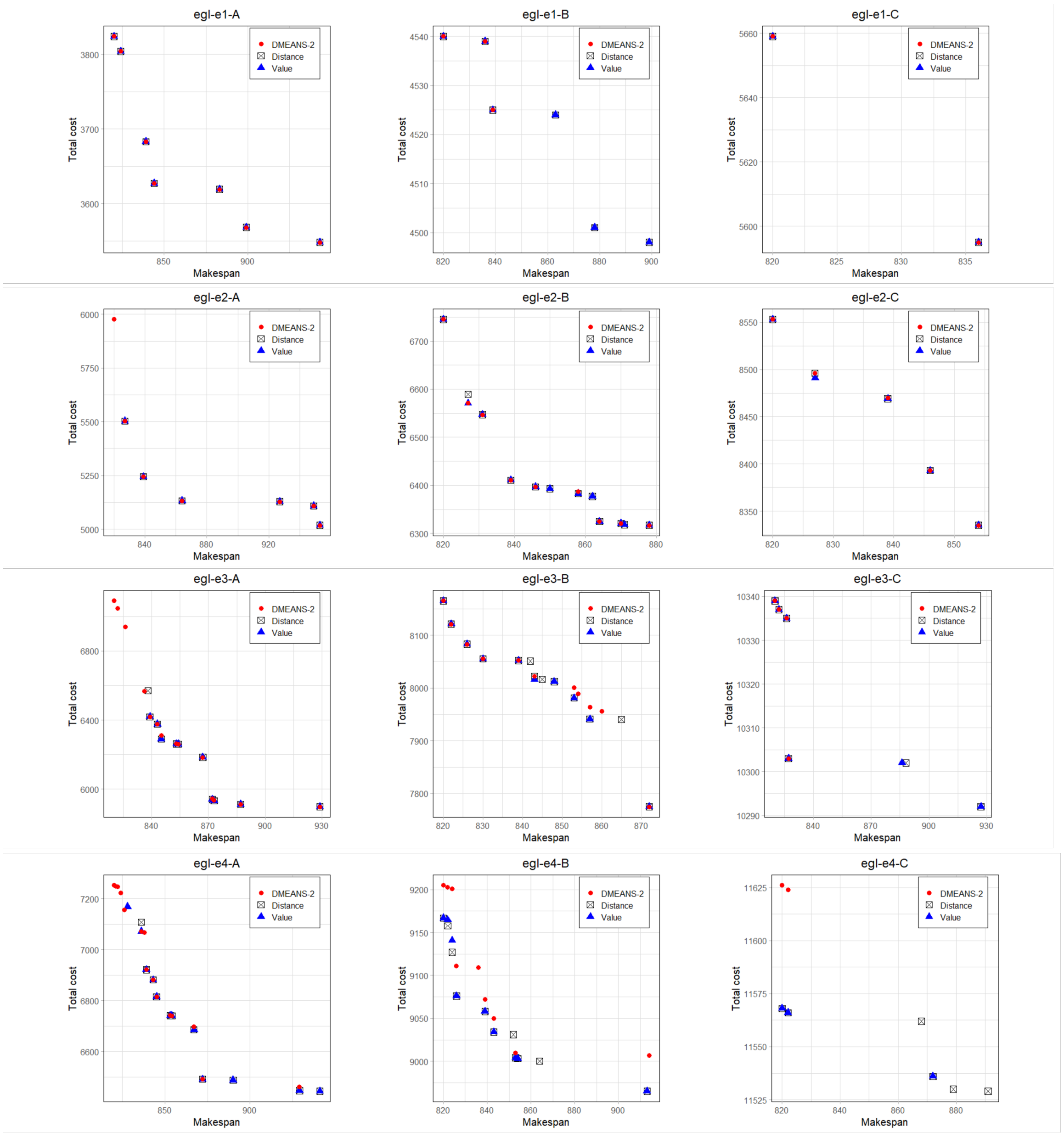


Figure 11: Pareto-front obtained by the proposed NSGA-II for various clone management strategies and the DMEANS-2 on the EGL instance.

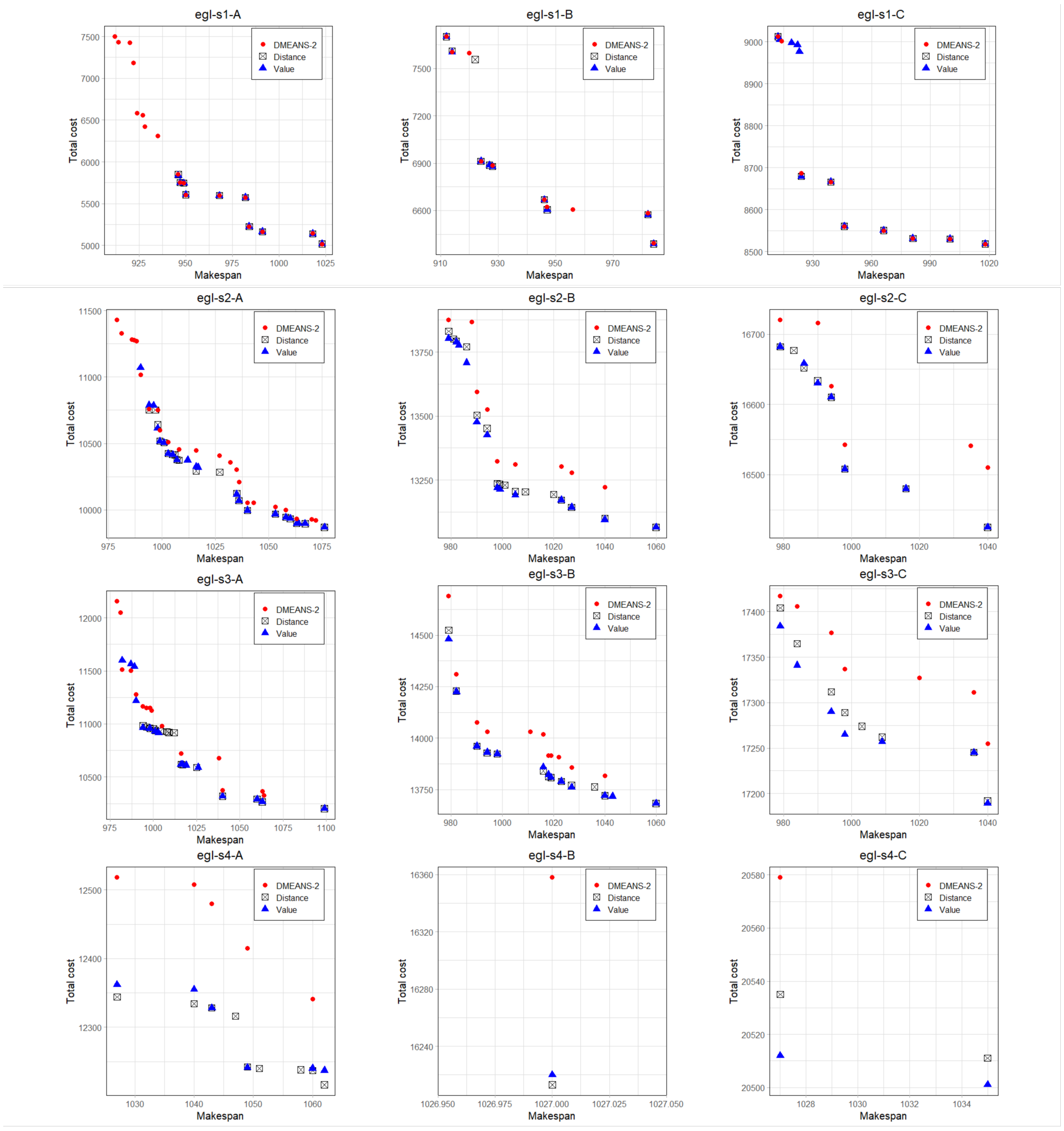


Figure 12: Pareto-front obtained by the proposed NSGA-II for various clone management strategies and the DMEANS-2 on the EGL instance.

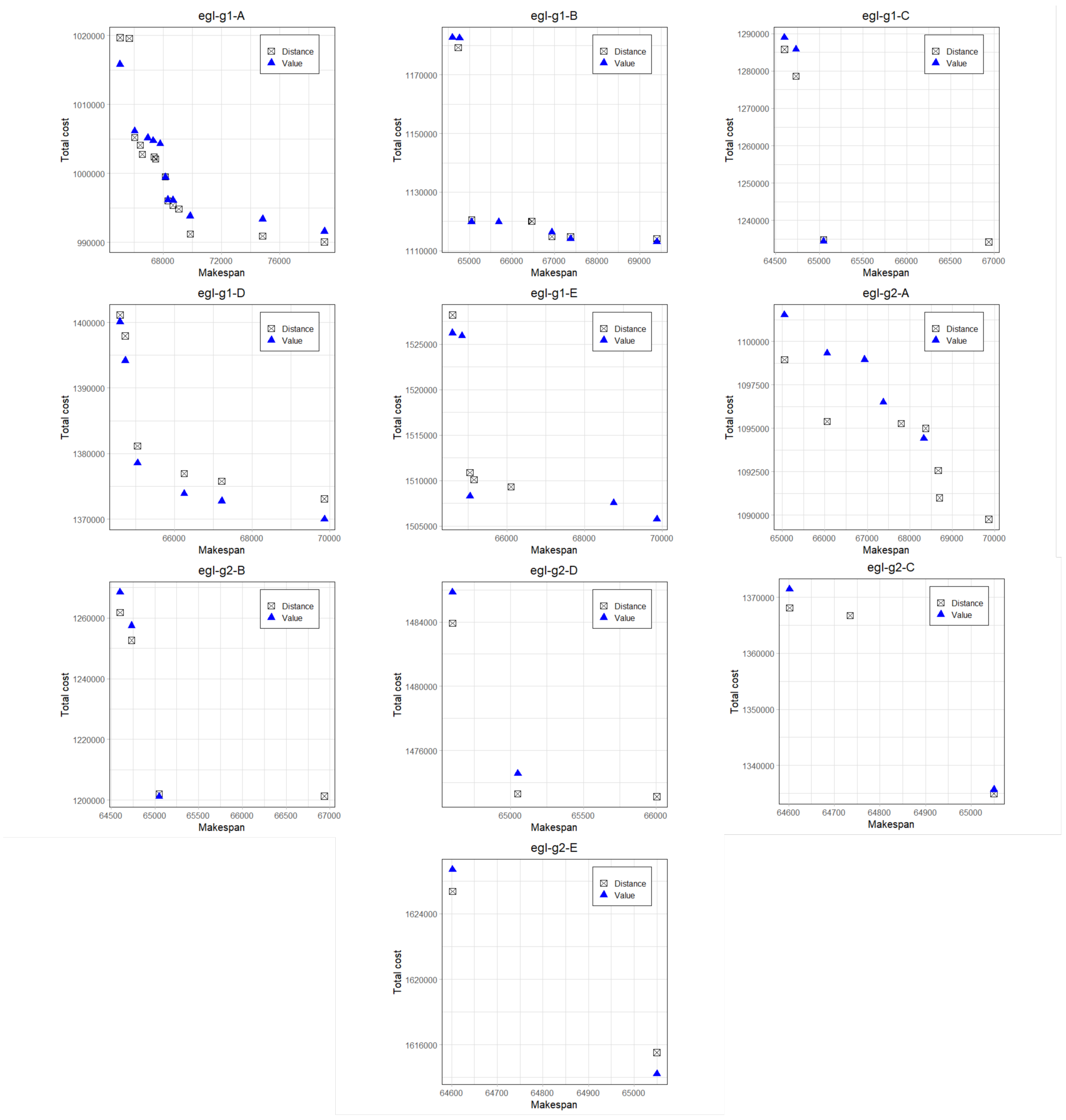


Figure 13: Pareto-front obtained by the proposed NSGA-II for various clone management strategies and the DMEANS-2 on the EGL-L instance.