Erasmus School of Economics

Bachelor Thesis Econometrie en Operationele Research

Quantitative Logistics and Operational Research

Investigating the performance of various PSO variants on VRPPD and VRPSPD

Abstract

This study considers different PSO variants, that have been previously studied in literature, to solve the vehicle routing problem with pickup and delivery and its simultaneous variant. We consider two PSO frameworks: GLN-PSO, and opposition based GLN-PSO. We combine these main algorithms with four existing solution representations SR-1, SR-1*, SR-2 and SR-P. Three of which use vehicle orientation points to construct feasible routes and one uses probabilities to construct feasible routes. Additionally, we propose a new solution representation SR-C, based on customer priorities and radii, and investigate its competitiveness. As opposed to previous research, we compare solution representations under similar PSO frameworks and local improvement procedures, resulting in a fair comparison. In this comparison there exists an important trade-off between computation time and solution quality. We conclude from the behaviour of the PSO methods that SR-2 and SR-1* are, generally, the best PSO methods depending on the specifics of the data instance. Additionally, the proposed SR-C is not competitive to the other solution representations.

Author:
Veroniek Visser

Supervisor:
MSc Y.N. Hoogendoorn
Second assessor:
Dr. K.S. Postek

July 7, 2019

ERASMUS UNIVERSITEIT ROTTERDAM

Contents

1 Introduction

The vehicle routing problem (VRP) is an extensively studied problem within the operations research field since its introduction by Dantzig and Ramser (1959). The standard VRP is defined as the construction of various routes, starting and ending at a single depot, for a homogeneous fleet of vehicles that must visit a group of customers. The objective of the problem can differ based on the needs of the solution. The VRP minimizes the number of vehicles with fixed cost f and travelled distance with variable cost g. In addition, many extensions of the standard VRP have been mathematically formulated.

In the capacitated VRP (CVRP), first introduced in Dantzig and Ramser (1959), trucks have a limited load capacity. Various mathematical formulations for this problem have been presented, such as the three index vehicle flow formulation in Golden, Magnanti, and Nguyen (1977). A two index flow formulation is presented in Laporte, Nobert, and Desrochers (1985). Additionally, a set partitioning formulation was proposed by Balinski and Quandt (1964). The advantage of the set partitioning formulation is its applicability to many of the extensions of the CVRP such as the ones defined below.

In the VRP with pickup and delivery (VRPPD) capacitated vehicles that can pickup and deliver goods from/to customers are considered. Some customers only have a pickup demand, which should be picked up by a truck at the customer and delivered to the depot. Other customers only have a delivery demand, that should be loaded at the depot and delivered to the customer. In the simultaneous VRPPD (VRPSPD), customers can have both a pickup and delivery demand. In Ai and Kachitvichyanukul (2009b), a three index mathematical flow formulation is provided which represents both the VRPPD and the VRPSPD.

Finding solutions for these problems is essential, as many large distribution companies deal with these problems every day. To indicate the size of the Dutch transportation industry: over 80% of Dutch domestic goods is transported by trucks and more than 100 million tonnes of Dutch goods are exported by trucks. As transportation by trucks has a tremendous influence on global warming through CO2 emissions, it is necessary to reduce the number of trucks and traveled distance of the used trucks as much as possible. Additionally, a decrease in traveling distance to transport goods enhances the profit of companies, as their costs decrease. Thus, it is of great importance to find good solutions for these problems.

An overview of exact methods to solve the VRP and its extensions is provided in Cordeau, Laporte, Savelsbergh, and Vigo (2007), but the problems have proven to be NP-Hard, as stated by Ai and Kachitvichyanukul (2009a). Literature has been focusing on developing heuristic approaches in the last decades. Heuristic methods for the VRP can be divided into two main categories: classical heuristics and meta-heuristics. Classical heuristics were the first to be developed in the period 1960-1990, whereas meta-heuristics have been proposed in the last decade and are still extensively investigated.

Meta-heuristics are partitioned into three subgroups as done in Cordeau et al. (2007): local search, population based and learning mechanisms. Population based algorithms, such as Wang and Chen (2012), generate a population of solutions for the VRP which are then manipulated into a new generation of solutions. The idea is that the solutions improve in each generation converging to good VRP solutions. Learning mechanisms, such as particle swarm optimization (PSO), presented in Kennedy and Eberhart (1995), use several learning structures to generate better solutions in each iteration.

PSO is a relatively new algorithm among the meta-heuristics for VRP, therefore little research has been done into its performance on VRP and especially its extensions. PSO considers particles with certain positions, which can be converted into VRP solutions based on the solution representation. This conversion is done with a decoding method, which often have local improvement heuristics incorporated. A PSO method is defined by its general algorithm (framework) and its solution representation. In literature various PSO frameworks have been defined to solve optimization problems in general, such as GLN-PSO Kachitvichyanukul, Purintrapiban, and Utayopas (n.d.) and opposition based PSO Rahnamayan, Tizhoosh, and Salama (2008). There are three solution representations proposed to solve the CVRP in Kim and Son (2012) and Ai and Kachitvichyanukul (2009a). In these papers, comparisons were made of the different solution representations, but different PSO frameworks and local improvement procedures where used in these methods. This may have resulted in unfair comparisons, such that we cannot conclude which PSO method performs best, based on previous research. Furthermore, extensive analysis of the performance of these variants compared to each other on the VRPPD and VRPSPD are missing from existing literature, as only one PSO method has been tested on VRPPD and VRPSPD instances by Ai and Kachitvichyanukul (2009b). Ai and Kachitvichyanukul (2009b) show that their PSO method is competitive compared to best known solutions to solve these problems. Thus, our research concentrates on the following research question:

How do different solution representation perform compared to each other when solving the VRPPD and VRPSPD in the GLN-PSO framework?

To answer this main research question, we have defined the following sub questions:

- Are the results presented in Ai and Kachitvichyanukul (2009b) replicable?
- How do various solution representations perform, when considering the same local improvement procedures?
- Can we enhance the PSO algorithms by implementing an opposition based initialization?

In Section 2, we provide a more extensive description of the PSO algorithm. Furthermore, in Section 3 we provide a detailed description of the solution representations that we consider in the different PSO methods. In Section 4, we describe the local improvement procedures that are used in the decoding methods. In Section 5, we describe the VRPPD and VRPSPD data used to compute the results, which are presented in Section 6. At last, we present our conclusions in Section 7.

2 PSO framework

In this section, we consider the particle swarm algorithm framework that we use to solve the VRP variants. The PSO framework itself does not obtain VRP solutions, but aims to optimize particle solutions, making it useful for optimization problems in general. The PSO framework is always used in combination with a certain solution representation, which is tailored for the optimization problem - in our case the VRP variants. In this study, we consider two variants of PSO: GLN-PSO and opposition based GNL-PSO. In Section 2.1, we provide a description of the GLN-PSO, after which we describe the opposition based GLNO-PSO (GLNO-PSO) variant in Section 2.2.

2.1 GLN-PSO

PSO is a evolutionary optimization method, which attempts to mimic the movements of swarm individuals to overcome obstacles and was first introduced by Kennedy and Eberhart (1995). In this study, we consider a variant of the standard PSO, namely the GLN-PSO. The GLN-PSO, proposed by Kachitvichyanukul et al. (n.d.), which incorporates more swarm effectiveness than the standard PSO.

In PSO, we consider a swarm of L particles, that by means of both social and cognitive learning behaviour, move within the search space of a problem towards a (local) optimum. This search space has a limited range, consisting of minimal positions ($\Theta^{\min} \in \mathbb{R}^H$) and maximal positions ($\Theta^{\max} \in \mathbb{R}^H$). Each particle has an associated position within the search space ($\Theta_l \in \mathbb{R}^H$) and velocity ($\Omega_l \in \mathbb{R}^H$). Each particle position can be decoded into a solution of the optimization problem, with a properly defined solution representation and decoding method. The solution representations we adopt and the corresponding decoding algorithms are discussed in Section 3. When a particles position is decoded, its corresponding fitness value ($Z(\Theta_l)$ can be calculated. This fitness value differs depending on the solution representation, but generally corresponds with the objective value of the VRP solution. Since VRP is a minimization problem, PSO aims to find particles with the lowest fitness value, implicating that a better particle has a lower fitness value.

As the velocity determines the movement of a particle, it influences the quality of solutions of the next iteration. Therefore, updating the velocity is dependent on the following three evolutionary (learning) factors.

1. Inertia: forces a particle to move in the same direction as the previous iteration, its weight (w) controls the impact of the current particles' velocity on the new particles' velocity.
2. Cognitive learning: forces a particle to move in the direction of its personal best position (Ψ_l^P), defined as best position an individual particle has found so far.
3. Social learning: forces a particle to move in the direction of the global best (Ψ^g), the best position that the swarm has found so far, in the case of the standard PSO. When considering the GLN-PSO, the social term additionally forces a particle in the direction of the local best (Ψ_l^L) the best Ψ_l^P among K neighbours of a particle and the near neighbour best (Ψ_l^N). An

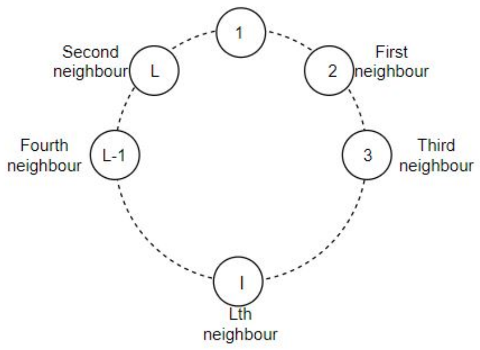


Figure 1: Neighbours of the first particle

interpretation of the latter two best positions is discussed below.

In the GLN-PSO, as described by Kachitvichyanukul et al. (n.d.), the K neighbours that are used to establish the Ψ_l^L, are determined using particle indices. The particles are structured in a ring shape, after which the neighbouring particles are determined. This concept of neighbourhood is displayed in Figure 1, with indications of the neighbours of the particle with index 1. Although this neighbourhood topology performs well, it does not include any spatial information from the particles. This, along with the fitness of the particles, is incorporated in the FDR topology that is used for generating Ψ_l^N. The near neighbour best position can be described as the position that maximizes the FDR as defined in (1) for each dimension.

$$FDR = \frac{Z(\Theta_l) - Z(\Psi_o)}{|\theta_{lh} - \psi_{oh}|} \text{ with } l \neq o \tag{1}$$

As PSO is an evolutionary algorithm, swarm diversity is an important factor, as noted by Shi and Eberhart (2008), some randomness is also incorporated in the particles velocities. This randomness is incorporated via random numbers u_p, u_g, u_l, u_n which are all random uniform numbers in the interval [0, 1]. However, it is not clearly specified by Ai and Kachitvichyanukul(2009b) or Kachitvichyanukul et al. (n.d.) whether these random numbers should be regenerated for each particle and/or dimension. We choose to regenerate the random numbers for each dimension and for each particle. Additionally, acceleration factors c_p, c_g, c_l, c_n are included in order to give different importance to different learning mechanisms.

A detailed GLN-PSO algorithm is provided in Algorithm 1. In this paragraph we explain the algorithm with references to the algorithm steps within brackets. We iterate T times, and in each iteration (τ) the particles move towards a new position based on their velocity. Then for each iteration, we evaluate the performance of a particles position $Z(\Theta_l)$ (5) and update the best positions accordingly (6-11). Based on the best positions and the current positions, the velocity of a particle is updated (13), via Equation (2) and then the new position of a particle is calculated (14). Finally, we must ensure that the new particles positions are within range (15-16).

$$\omega_{lh}(\tau+1) = w(\tau)\omega_{lh}(\tau) + c_p u_p(\psi_{lh}^P - \theta_{lh}(\tau)) + c_g u_g(\psi_h^G - \theta_{lh}(\tau)) \\ + c_l u_l(\psi_{lh}^L - \theta_{lh}(\tau)) + c_n u_n(\psi_{lh}^N - \theta_{lh}(\tau)) \quad (2)$$

In the following subsections we provide more information on procedures in the PSO that are adapted when considering other frameworks or solution representations.

Algorithm 1 General PSO algorithm

```
1: Initialize a swarm of $L$ particles with random positions and all velocities equal to zero.
2: $\Psi_l \leftarrow \Theta_l$ for each particle $l$
3: for $\tau = 1 \dots T$ do
4:     for $l = 1 \dots L$ do
5:         Decode the position into a VRP solution $R_l$ and compute its corresponding fitness $Z(\Theta_l)$
6:         if $Z(\Theta_l) \leq Z(\Psi_l)$ then $\Psi_l \leftarrow \Theta_l$
7:         if $Z(\Psi_l) \leq Z(\Psi_g)$ then $\Psi_g \leftarrow \Psi_l$
8:         Set $\Psi_l^L$ to lowest personal best among K neighbours
9:     for $l = 1 \dots L$ do
10:        for $h = 1 \dots H$ do
11:            $\psi_{lh}^N \leftarrow \psi_{oh}$ for the particle $o = 1 \dots L$ that maximizes the FDR in Equation (1)
12:            $w(\tau) = w(T) + \frac{\tau - T}{1 - T}(w(1) - w(T))$
13:            Update $\omega_{hl}$ according to Equation (2)
14:            $\theta_{lh}(\tau+1) = \theta_{lh}(\tau) + \omega_{lh}(\tau+1)$
15:            if $\theta_{lh}(\tau+1) \geq \theta^{\max}$ then $\theta_{lh}(\tau+1) = \theta^{\max}$ and $\omega_{lh}(\tau+1) = 0$
16:            if $\theta_{lh}(\tau+1) \leq \theta^{\min}$ then $\theta_{lh}(\tau+1) = \theta^{\min}$ and $\omega_{lh}(\tau+1) = 0$
```

2.2 GLNO-PSO

In opposition based GLN-PSO (GLNO-PSO), an extra step is included when initializing the particle swarm, the remainder of the algorithm remains the same as the GLN-PSO. After initialization of the particles $l = 1 \dots L$, we construct the opposite positions of these particles through Equation (3). When these extra particles are constructed and the fitness values of all particles in the swarm (that has size $2L$) are computed, we chose the L particles that have the lowest fitness value. This extra initialization step has been studied on the standard PSO by Rahnamayan et al. (2008). They have shown that it can enhance the optimization procedure of the standard PSO, considering the convergence rate and the solution quality. To our knowledge, opposition based initialization has never been used together with GLN-PSO in optimization problems.

$$\theta_l^o = \theta_{\min} + \theta_{\max} - \theta_l \quad (3)$$

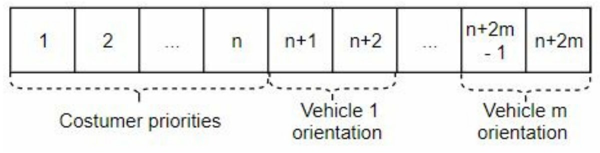
3 Solution representations and decoding methods

As mentioned in the previous section, the solution representation is an important element of the PSO, after all the particle position only becomes a solution to the optimization problem after decoding it. In this section, five solution representations and corresponding decoding methods are discussed: SR-1, SR-1*, SR-2, SR-C and SR-P. In order to improve the solutions generated by the decoding method, three solution improvement methods are described.

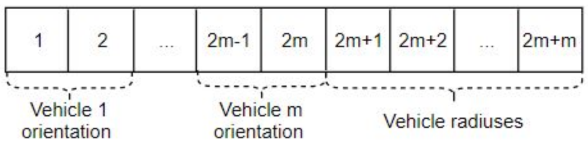
3.1 SR-1 and SR-1*

SR-1 was first used to solve the VRPSPD by Ai and Kachitvichyanukul (2009b) and it has shown to be competitive to best known solutions. It was introduced using only one local improvement technique in the decoding procedure. For comparison purpose, we also introduce SR-1*, which is the same as SR-1, but uses all three local optimization procedures as described in Section 4 in the decoding method. As SR-1* uses more local optimization techniques than SR-1, we know that SR-1* should always perform at least as good as SR-1 or even better.

In SR-1, a particle consists of $2n+m$ dimensions. Here each of the first n dimensions is assigned to an individual customer. Then, the value of the dimension represents the routing priority of the customer, where a lower value indicated a higher priority to insert a customer into a route. Thus, from this first n dimensions, we can construct the customer priority list U. The other $2m$ dimensions are used to construct m vehicle orientation points. Here, the x-coordinate of the point in the service map is represented by a dimension and the y-coordinate is represented by another dimension. In Figure 2a, a visual representation of a particle with its dimensions is displayed. A vehicle orientation



(a) Particle dimensions in the SR-1 Representation



(b) Particle dimensions in the SR-2 Representation

Figure 2: Particle dimensions visualizations

point indicates the area where the vehicle will most likely serve its customers. Therefore a vehicles route will aggregate around its orientation point. In the left part of Figure 3, this idea is visualized. The thickness of the dashed lines indicate the priority of that customer for that vehicle. Using the vehicle orientation point, the vehicle preferences of customers can be computed in a vehicle priority matrix W. A customer prefers vehicle a over vehicle b, if the Euclidean distance from the customer to vehicle a is smaller than the distance of that customer to vehicle b. This ensures that customers that are close to each other on the service map, are included in the same route. Using both U and W, we can compute a VRP-solution. The detailed description of the decoding method is included in Algorithm 2. Using this algorithm, it may happen that a customer is not inserted. Therefore we compute the fitness value of a particle as the objective value plus a large penalty, based on the number of uninserted customers.

3.1.1 Appropriate number of vehicles

Solution representations SR-1 and SR-2 are based on a fixed number of vehicles, also in the decoding methods the number of vehicles m is not subject to change. Therefore Ai and Kachitvichyanukul

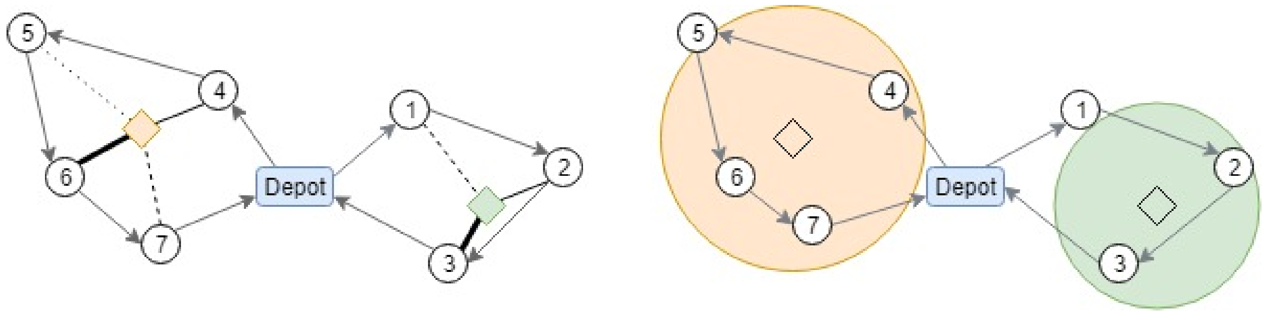


Figure 3: Vehicle routes with vehicle priority and service area. The customers are represented as white circles whereas the vehicle orientation points are diamond shaped.

Algorithm 2 Decoding method SR-1 or SR-1*

```
 1: Construct the priority list of customers
 2: Construct the vehicle priority matrix $W$ by using the Euclidean distance from each customer to the vehicle orientation points
 3: procedure CONSTRUCT VEHICLE ROUTES
 4:     Set $k \leftarrow 1$
 5:     repeat to add customers one by one to a route
 6:         Set $c \leftarrow U_k$, and $b \leftarrow 1$
 7:         repeat to make a candidate for a new route
 8:             Set $j \leftarrow W_{cb}$
 9:             Insert $c$ to route $R_{lj}$ via cheapest insertion heuristic
10:             if $R_{lj}$ is feasible then Re-optimize with 2-opt method and $b \leftarrow m$
11:             else $b \leftarrow b + 1$
12:         until $b = m$
13:         $l \leftarrow k + 1$
14:     until $k = n$
15:     Only SR-1*: improve the routes by using 1-1 exchange and 1-0 exchange
```

(2009b) propose a method to search for an appropriate number of vehicles to be used by the particles of the PSO algorithm. The algorithm that they propose is included in Algorithm 3.

Algorithm 3 Set number of vehicles

```
1: Generate a random particle of n+2m dimensions and set v=m
2: Decode the particle using Algorithm 2 and compute the fitness value Z
3: Remove the dimensions of the vehicle that serves the least number of customers from the particles position
4: set v=v-1
5: Decode the particle using Algorithm 2 and compute the fitness value Z'
6: if $Z' < Z$ then
7:     Go back to Step 3
8: elseAdd back the dimensions last removed and set m=v+1
```

3.2 SR-2

This second solution representation was first proposed by Ai and Kachitvichyanukul (2009a) and was applied to the CVRP. In this paper, SR-1 and SR-2 were both tested on the same CVRP instances. Results showed that SR-2 computed better VRP solutions, when only taking travelled distance into account. However, the computational time of the SR-2 representation was higher. SR-2 uses all local optimization procedures, as described in Section 4, which is an advantage compared to SR-1. This may also have resulted in more computational time.

The SR-2 representation is an extended version of the SR-1 representation and exploits the idea of customer proximity in routes further. In this representation, the particles are formed with $3m$ dimensions. Here, the first $2m$ dimensions form the vehicle orientation points, like the last $2m$ dimensions in the SR-1 representation. The last m dimensions represent the vehicle coverage radius. The vehicle coverage area is the circle with as center the vehicle orientation point and as radius the corresponding vehicle coverage radius. This is visualized in the right part of Firgure 3, where the orange and green circles represent vehicle coverage areas. In Algorithm 4, the decoding method is described in detail.

Algorithm 4 Decoding method for SR-2

```
 1: for $j = 1 \dots m$ do
 2:     $xref_j \leftarrow \theta_{i,3j-2}$ and $yref_j \leftarrow \theta_{i,3j-1}$
 3:     $r_j \leftarrow \theta_{i,3j}$
 4: for each customer do
 5:     Insert the customer in the route of the nearest vehicle orientation point with cheapest insertion
 6:     Check feasibility of the route, taking into account that a customer should be in the vehicle coverage area of the vehicle
 7: Optimize the constructed routes by using 2-opt, 1-1 exchange and 1-0 exchange
 8: for Any remaining customers do
 9:     Insert customer located furthest away from the depot
10:     Insert the customer to the closest vehicle that has not been tried before
11:     if Constructed route is infeasible then
12:         Remove customer and go back to 10
13: Optimize the constructed routes by using 2-opt, 1-1 exchange and 1-0 exchange
```

3.3 SR-C

In this section, we introduce a new solution representation which is inspired by SR-1 and SR-2. SR-C extends the idea of the customer priority list from SR-1, but it additionally considers customer radii. The particles consist of $2n$ dimensions, of which the first n are the same as in SR-1. The second n dimensions represent the customer radii. Addition areas around each customer can be constructed as circles around the customers coordinate with the radius that corresponds to the customer. With the addition areas and the customer coordinates, we can construct the addition area matrix. For each customer c, we define a row vector of customers whose coordinates lie within c's addition area.

The description of the SR-C decoding method is presented in Algorithm 5. As in SR-1, we insert customers one-by-one based on their priority. In SR-C, a customers priority is not only based on the first n dimensions of the particle, but also on the customer radius. A customer is only added to a route if it lies within the addition area of any other customer that has already been inserted (into any route). If this does not result in any feasible routes, we try to insert the customer to a route of which any customer lies within his addition area. If this also does not result in a feasible route, we open a new route and add the customer to it. The advantage of SR-C over SR-1 and SR-2 is that it does not set the number of vehicles in the first iteration, but the number of vehicles in the solution is determined in the decoding method. On the other hand, this may even well be a disadvantage as it may lead to the use of many vehicles.

Algorithm 5 Decoding method for SR-C

```
 1: for $j = 1 \dots m$ do
 2:     $p_j \leftarrow \theta_{i,j}$ and $r_j \leftarrow \theta_{i,2n+j}$
 3: Sort the customers based on their value of $p_j * r_j$ in list $L$
 4: Construct the addition area matrix $A$
 5: for each customer $c$ in $L$ do
 6:     currentroute = $A[c][b]$
 7:     repeat Insert customer into currentroute through cheapest insertion
 8:         if Route is feasible then Maintain route and perform 2-opt
 9:         else Remove customer from currentroute and $b \leftarrow b + 1$
10:     until Feasible route is found or b exceeds the size of A
11:     currentroute = $A[b][c]$
12:     repeatInsert customer into current route through cheapest insertion
13:         if Route is feasible then Maintain route and perform 2-opt
14:         elseRemove customer from currentroute and $b \leftarrow b + 1$
15:     until Feasible route is found or b exceeds the size of A
16: Optimize the constructed routes by using 2-opt, 1-1 exchange and 1-0 exchange
```

3.4 SR-P

In this section, we provide a description of the probability based solution representation SR-P. As this solution representation is probability based, there are some extra restrictions we have to take into account. Therefore, we also introduce some adaptations that are made to the PSO framework in Section 3.4.2.

3.4.1 Solution representation description

SR-P is proposed by Kim and Son (2012) in combination with standard PSO to solve the CVRP and uses particles of $(n + 1) \times (n + 1)$ dimensions. They have implemented this solution representation using an extra local improvement procedure, which for comparison purposes we do not implement. SR-P has shown to be competitive compared to SR-1 and SR-2 (Kim & Son, 2012). The first dimension represents the depot and the other n dimensions, portray the customers. Each particles' position represents a probability matrix with entries p_{ij}, which indicate the probabilities of connecting node i to node j in the decoding method. As we deal with a probability matrix, the following conditions should hold for the matrix entries:

$$\sum_{j=1}^{n} p_{ij} = 1 \quad \text{for } i = 1 \dots n+1, \tag{4}$$

$$p_{ij} \in [0, 1] \quad \text{for } i = 1 \dots n+1 \text{ and } j = 1 \dots n+1.$$

When decoding a particle from the probability matrix to a VRP solution, we consider Algorithm 0 developed by Kim and Son (2012).

3.4.2 Adaptations to the PSO framework to use SR-P

The initialization of the optimization method is not performed randomly, as this does not result in particles satisfying conditions from Equation (4). We rather perform multiple sweep heuristics

Algorithm 6 Decoding algorithm for SR-P

```
 1: repeat
 2:     Set the depot as start of the route
 3:     Select the next customer on the route based on the probabilities of the previous node
 4:     Add this customer to the route by adding it to the previous node
 5:     if Route is feasible then
 6:         Maintain the route
 7:         Remove customer from probability matrix and normalize the new matrix
 8:     else Return to the route from step 3
 9:         And connect last customer to the depot node
10: until All customers are added to a route
11: Perform 2-opt, 1-0 exchange and 1-1 exchange to the solution
```

to obtain a set of L feasible VRPPD or VRPSPD solutions. We use the sweep algorithm adapted from Clarke and Wright (1964), that is also implemented by Kim and Son (2012).

Algorithm 7 Sweep algorithm

```
1: Sort the customers in a counter clockwise manner based on their polar angle with the depot in list $L$
2: Chose a random index of $L$ and rearrange $L$ starting at this index
3: repeat
4:     Start a new route
5:     Assign the first unrouted customer from the list to the route.
6:     Select the nearest customer from the current route. Continue to assign such nearest customers to construct the route
       while the sum of customer demands does not exceed the capacity of the route
7: until All customers have been asigned a route
8: Perform 2-opt, 1-0 exchange and 1-1 exchange to solution
```

After the sweep algorithm is performed, we encode the VRP solutions to particles positions. In the encoding method, we consider one node at a time k and determine the set of arcs C that are connected to it, independent of the direction of the arc, in the solution. We construct the probabilities to another node r as $P_{kr} = \frac{I_r}{|C|}$, where I_r is an indicator function that is value one if r is in the set of connected arcs to k, C and zero otherwise. From Figure 3, we can construct the following particle using the encoding method

$$\begin{bmatrix} 0 & 0.25 & 0 & 0.25 & 0.25 & 0 & 0 & 0.25 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \end{bmatrix}$$

Furthermore, Kim and Son (2012) propose to update the particles position only through random weights and not depending on velocities, as this lead to better results. Thus, the following formula is used to determine a vehicles next position

$$\Theta_l(\tau+1) = w_1(\tau)\Theta_l(\tau) + w_2(\tau)\Psi_l^P(\tau) + w_3(\tau)\Psi^G(\tau) + w_4(\tau)\Psi_l^L(\tau) + w_5(\tau)\Psi_l^N(\tau). \tag{5}$$

We chose the coefficients $w_2 \dots w5$ randomly such that their sum equals 1, as this ensures that the new position satisfies (4) if all best position matrices also satisfy these constraints. By construction, only the near neighbour best position does not necessarily satisfy the constraints, therefore it should be normalized such that the constraints are satisfied. The first coefficient is as normal the predefined inertia weight, the other weights are randomly chosen. We choose $w_2 \dots w_5$ in the following manner. First, we chose w_2 as a random number in the range $[0, 1 - w_1]$. Then we select w_3 randomly from the interval $[0, 1 - w_1 - w_2]$ and w_4 from the interval $[0, 1 - w_1 - w_2 - w_3]$. Finally, we set $w_5 = 1 - w_1 - w_2 - w_3 - w_4$.

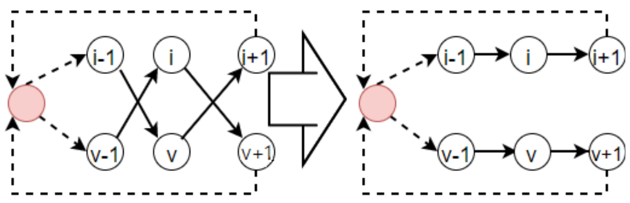
We choose to implement the near neighbour best, optimizing the FDR for each column of the particles dimension. As we cannot calculate the absolute value of two vectors in the denominator, we use the Euclidean distance instead.

4 Solution improvement methods

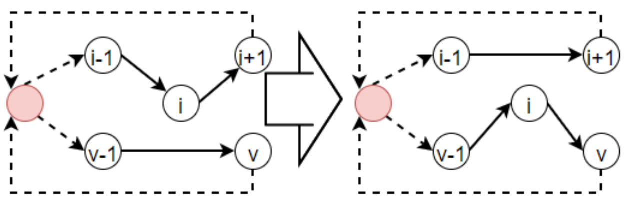
As the decoding methods are heuristic methods to construct routes from the dimensional particles, they may be further improved with local improvement methods. These methods make small adjustments, such as interchanging two customers in a route, in order to optimize the constructed solution. In the decoding phases, three of such methods are used: 2-opt, 1-0 exchange and 1-1 exchange. The 2-opt method aims to improve routes by only considering changes within a single route, whereas the exchange methods take two routes to look for improvement. In Algorithms 8 and 9, the algorithms for these methods are provided and in Figures 5, 4a and 4b visual representations of the methods are shown.

Algorithm 8 1-1 exchange or 1-0 exchange

```
1: Set $n=$ number of customers in first route and $m$ =number of customers in second route
2: for $i = 1 \dots n$ and $j = 1 \dots m$ do
3:     Modify route by interchanging two customers, as in Figure 4b for 1-1 exchange or as in Figure 4b for 1-0 exchange
4:     if New route is feasible and has lower costs then
5:         Maintain modified route
6:     else Undo changes made in 3
```



(a) Illustration of a 1-1 exchange procedure



(b) Illustration of a 1-0 exchange procedure

Figure 4: Illustrations of local optimization procedures

Algorithm 9 2-Opt procedure

```
1: Set $n$ = number of customers in the route
2: for $i = 1 \dots n-2$ and $j = i+2 \dots n$ do
3:     Change route as in Figure 5
4:     if New route is feasible and has lower costs then
5:         Maintain modified route
6:     else Undo changes made in 3
```

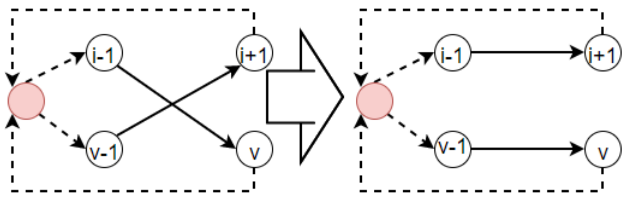


Figure 5: Illustration of a 2-opt procedure

5 Data

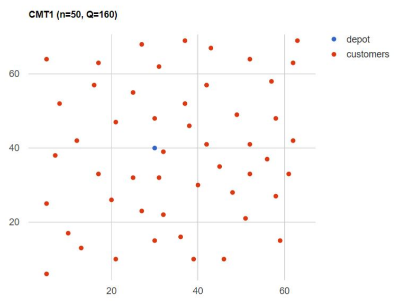
In this section we describe the data instances that we use to compute our results. We use the data that was first introduced by Nagy and Salhi (2005) and consist of five problem sets: CMTH, CMTT, CMTQ, CMTX and CMTY. The data sets, consisting of 14 instances each, provide us with customer indices, pickup demand, delivery demand, vehicle capacity and the Cartesian coordinates of the depot and customers. Additionally some instances (6-10 and 13, 14) consider a duration limit and service times at customers. For these instances, the total travelled distance of the routes together with the total service times cannot exceed this duration limit.

The data sets are based on the Christofides data instances CMT1-CMT14, the pickup and delivery demands were adapted following the methods of Nagy and Salhi (2005). We note in the coordinates of the data sets that in CMT instances 11 through 14 the customers are clustered in the map, as is shown in Figure 6b. We expect that SR-1* and SR-2 perform better than SR-P and SR-C, because the vehicle orientation points suit these clustered instances best.

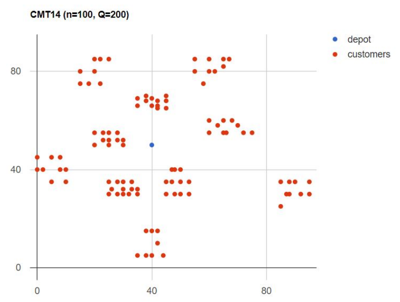
In data sets CMTH, CMTT and CMTQ the customers only have a pickup demand *or* delivery demand, thus we use these instances as VRPPD instances. In set T, only 10% of the customers have a pickup demand and in sets Q and H this is respectively 25% and 50%. When constructing these data sets, we take the CMT instances and change every tenth, fourth and second customers delivery demand into a pickup demand. In instances X and Y, customers have a pickup and delivery demand, making these instance suitable for the VRPSPD problem.

Based on these instance characteristics, we divide each data set (CMTT, CMTH, CMTQ, CMTX and CMTY) into six subgroups

- SNU: (1-3) instances with at most 100 unclustered customers, which have no duration limit;
- LNU: (6-8) instances with more than 100 unclustered customers, which have no duration limit;
- SDU: (4-5) instances with at most 100 unclustered customers, which have a duration limit;
- LDU: (9-10) instances with more than 100 unclustered customers, which have a duration limit;
- SC: (12, 14) clustered customer instances with at most 100 customers;
- LC: (11, 13) clustered customer instances with more than 100 customers.



(a) Instance CMT1



(b) Instance CMT14

Figure 6: Coordinate grid of customers in two instances. On the right, clustering is clearly visible.[1]

[1]From: http://vrp.galgos.inf.puc-rio.br/index.php/en/

6 Results

We have described a multitude of PSO methods that we perform on the VRPPD, VRPSPD and VRPSDPTW problems, therefore we have summarized the proposed PSO algorithms and solution representations, together with how they are referenced in the main text, in Table 1.

SR	PSO variant	Method reference
SR-1	GLN	SR-1
SR-1*	GLN	SR-1*
SR-2	GLN	SR-2
SR-P	GLN	SR-P
SR-C	GLN	SR-C
SR-1*	GLNO	obSR-1*
SR-2	GLNO	obSR-2
SR-C	GLNO	obSR-C

Table 1: Overview of the different PSO methods and their references in text

Parameter	Value	New value
Number of particles	50	50
Number of iterations	1000	1000
Number of neighbours	5	5
Inertia weight at first iteration	0.9	0.9
Inertia weight at last iteration	0.4	0.4
c_p	1	0.5
c_g	0	0.5
c_l	1	0
c_n	2	1.5

Table 2: Overview of the PSO original PSO parameters and the addapted PSO parameters

As was done in Ai and Kachitvichyanukul (2009b), we do not minimize the number of vehicles in the solutions, setting the fixed costs for the vehicles to be 0 and the variable cost g equal to 1. Thus, the objective value of the solutions equals the total travelled distance. Furthermore, for the instances that take a duration limit into account, we set the time to travel from one customer to another to equal the distance between these customers.

In order to compute the results, we set the parameters for the PSO algorithm as presented in Table 2, which are the same as used by Ai and Kachitvichyanukul (2009b). Although the performance of PSO algorithms is dependent on the exact parameter settings, choosing the best performing parameters, taking the current problem types into account, would be a study on its own and has not been studied in literature. Ai and Kachitvichyanukul (2009a), also compare SR-1 and SR-2 for the CVRP, using the same parameter settings for both solution representations. Therefore, we choose to use the same parameters for each method. However, it is important to note that these parameter settings could be optimized in order to better compare the solution representations. After running preliminary experiments, we found that the solution representations do not perform as expected which is described in detail in Section 6.1. We decided to compute the results of SR-1*, SR-2, SR-P and SR-C with the parameter settings shown in Table 2.

For SR-1 and SR-2, we need to define a certain number of vehicles at the start. As done by Ai and Kachitvichyanukul (2009b), we set the number of vehicles m equal to the number of vehicles used in the best found solution until now. As no complete overview of the best found solutions of these instances is provided in literature, we consider the best found solutions as provided by Ai and Kachitvichyanukul (2009b).

We implement the described algorithms using Java in Eclipse. We used a computer with processor AMD PRO A4-8350B R5, 16 GB RAM and 3.5 GHz clock rate. For each method, we replicate the procedure ten times.

6.1 Comparison to AI results of SR-1

To ensure that we have implemented a competitive PSO algorithm, we first compare our results to the results found in Ai and Kachitvichyanukul (2009b). In Table 3 we report our average costs for the CMTT, CMTH, CMTQ, data sets together with the average results of computed by Ai and Kachitvichyanukul (2009b). Additionally, we include the deviation of the results compared to the results of Ai and Kachitvichyanukul (2009b) in %. We compute this deviation as follows: Deviation $= \frac{\text{Computed costs}-\text{Costs AK}}{\text{Costs AK}} \times 100\%$.

We include our results per instance in Appendix A for the VRPPD instances and in Appendix B for the VRPSPD instances.

Data instance	Costs (€) AK	Costs (€) SR-1	Deviation	Costs (€) SR-1^0	Deviation (%)
CMTH	908.5	1022.2	12.5	940.2	3.5
CMTT	950.3	1013.8	6.7	957.5	0.8
CMTQ	984.9	1048	6.4	995	1
CMTX	934.5	1000.6	7.1	945.5	1.2
CMTY	933.4	963.7	3.2	908.2	-2.7

Table 3: Average results for VRPPD and VRPSPD instances

We observe that the costs obtained by Ai and Kachitvichyanukul (2009b) are on average lower than our results. This difference may be caused by different parameter settings. As mentioned in Section 2.1, Ai and Kachitvichyanukul (2009b) did not specify whether the random numbers, used to update the velocity in Equation 2, should be different for each particle and/or dimension. Therefore, we varied the moment of random number generation in preliminary experiments, but this did not yield any improvements. Furthermore, Ai and Kachitvichyanukul (2009b) did not provide a detailed description on how to obtain the local best of a particle. It may be possible that we implemented this learning structure differently, using the implementation described in Kachitvichyanukul et al. (n.d.). Therefore, we varied the parameter settings in preliminary runs, setting them equal to the new values from Table 1. We refer to SR-1 with these parameter implementation as SR-1^0. In Table 3, we report the average best costs over ten runs of this PSO method. We observe that the SR-1^0 results are more similar to the results from Ai and Kachitvichyanukul (2009b). In Table 4, the results of some individual data instances are reported from AI, SR-1 and SR-1^0.

Data instance	Costs (€) AI	Costs (€) SR-1	Deviation (%)	Costs (€) SR-1^0	Deviation (%)
CMT1Q	490	494.2	0.9	489.7	-0.1
CMT3H	701	785.4	12.0	769.7	9.8
CMT6H	557	567.6	1.9	558.7	0.3
CMT14T	846	911.1	7.7	839.7	-0.7
CMT2X	710	740.5	4.3	707.5	-0.4
CMT3Y	740	748.1	1.1	723.7	-2.2
CMT8Y	902	971.9	7.7	904.2	0.2
CMT11X	895	940.1	5.0	927.1	3.6

Table 4: Representative solutions

We see that for most instances, the SR-1^0 costs deviate less than 1% from the results presented by

Ai and Kachitvichyanukul (2009b).

6.2 Performance analysis

In Appendix C (D), we include the best results over ten runs for the VRPPD (VRPSPD) instances, obtained by the various PSO methods. We present the average costs of the data sets in Table 5.

	Costs (€)						
Data instance	SR-1*	obSR-1*	SR-2	obSR-2	SR-C	obSR-C	SR-P
CMTH	920.3	918.4	913	907.1	1033.3	996.2	972.2
CMTT	985	985.3	977.7	979.1	1098.3	1103.8	1030.6
CMTQ	956	951.9	943	943.4	1035.8	1064.5	998.6
CMTX	932.7	932	920.7	926.1	1046.1	1037.6	999.4
CMTY	902.4	901.2	897	896.2	971	987.2	961.2

Table 5: The average best results of the various PSO methods for the VRPPD and VRPSPD instances

We observe that on average SR-2 with opposition based GLN-PSO and SR-2 with GLN-PSO perform best. obSR-1* and SR-1* follow the performance of SR-2 closely, but SR-P and (ob)SR-C do not present competitive results at all. Furthermore, we do not observe a clear difference between the GLN-PSO implementation and the GLNO-PSO implementation, when considering the same solution representation. This may suggest that there exists little to no influence of the opposition based initialization on the PSO methods.

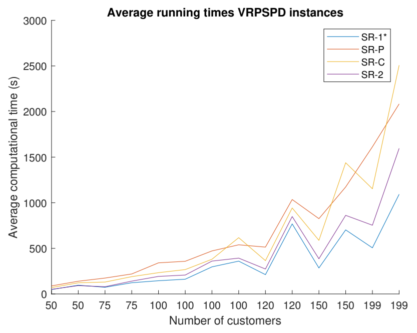
To analyze the individual performance of the PSO methods, we provide some representative individual results in Table 6.

	Costs (€)						
Data instance	SR-1*	obSR-1*	SR-2	obSR-2	SR-C	obSR-C	SR-P
CMT1H	465.1	465.3	465.9	466.5	465.1	465.3	533
CMT9T	1230	1224.3	1212.5	1210.3	1658.1	1443.5	1297.5
CMT3X	737.3	740.1	740.3	739.4	783.1	765.4	754.5
CMT8Q	876.8	871	865.5	869.9	914.1	969.5	917.6
CMT14Y	825.9	835.3	822.2	823.4	824.9	825.1	834.5
CMT10X	1491.2	1491.4	1460.6	1470.1	2062.2	2213.5	1670.1
CMT13Q	1564.1	1560.6	1549.8	1556.9	1693.5	1621.8	1583.6

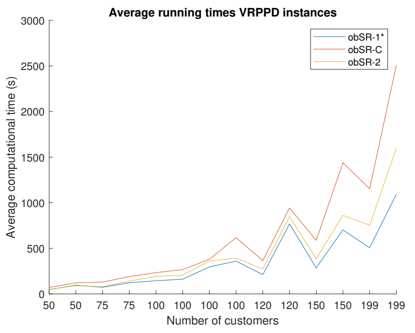
Table 6: Some representative best results of the various PSO methods for the VRPPD and VRPSPD instances

We note from Table 6. that SR-2 and obSR-2 perform best on the individual instances. Furthermore. we see that SR-C does provide reasonably well results for smaller customer instances, but is outperformed on the larger customer instances. Also, we observe that SR-P is not competitive to SR-1* or SR-2 on individual instances, which seems contradictory to previous results presented by Kim and Son (2012). We analyze these in more detail in Section 6.4.

In addition to the best found solution, the computational time of a method is also an important performance indicator. In Figure 7a, we present the average running times of the GLN-PSO methods for the VRPSPD instances. In Figure 7b, we present the average running times of the GLNO-PSO methods for the VRPSPD instances.



(a) Average running time of the PSO methods



(b) Average running time of the PSO methods

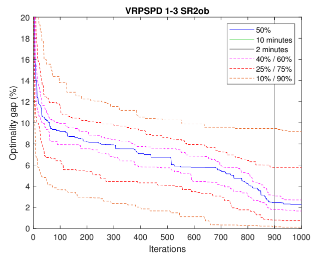
We observe that the average running times are increasing for the number of customers considered. Furthermore, we observe that the running times of GLN-PSO and GLNO-PSO hardly differ from each other. This is expected, as one extra step in the initialization is made when performing GLNO-PSO compared to GLN-PSO. Additionally, we clearly observe that for large customer instances SR-2 and SR-P are much slower compared to SR-1*. This is an important feature of an optimization method, as in the real world solutions have to be provided within reasonable amount of time. SR-2, SR-P and SR-C seem to have an quadratic or exponential relation with instance size. In Ai and Kachitvichyanukul (2009b), it is noted that the running time for the SR-1 solution representation is linear. When considering SR-1*, we observe that the relation of the computational time and instance size resembles more a quadratic function. This implies that the addition of the extra local improvement procedures have decreased the computational speed.

6.3 Behaviour analysis

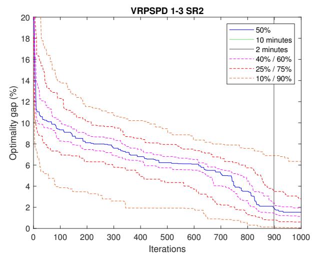
To extend the performance analysis, we look at the performance of the algorithm within its 1000 iterations. We refer to this as the behaviour of a PSO method. We perform this analysis on subgroups of the data sets, which were previously defined in Section 5.

For each of the groups, we aggregate the optimality gap to our best computed solution, at every iteration, for each run of a PSO method. We compute figures with quantiles of the optimality gaps at every iteration.

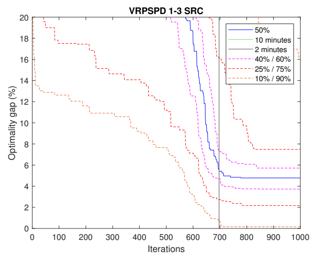
In preliminary runs, we did not observe behavioural differences between the three (two) different VRPPD (VRPSPD) instances. Therefore, we chose to aggregate the VRPPD (VRPSPD) subgroups with each other. With these figures, we can compare the different methods in the average case, but also the worst and best case scenarios can be compared. In Figure 8, we included the behavioural plots of the SNU subgroup for the different PSO methods.



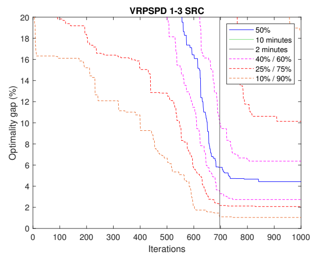
(a) obSR-2



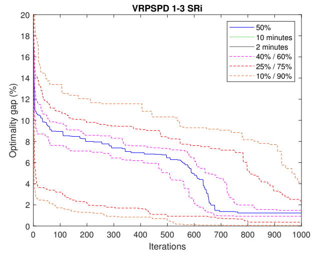
(b) SR-2



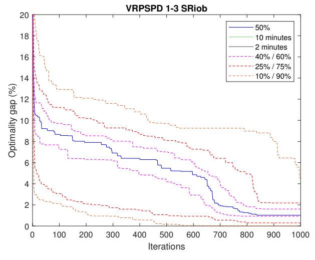
(c) obSR-C



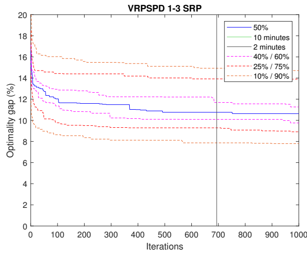
(d) SR-C



(e) SR-1*



(f) obSR-1*



(g) SR-P

Figure 8: Development of optimality gap during 100 iterations, where all runs of the instances 1-3 of CMTQ, CMTT and CMTH (SNU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

In Figure 8, we clearly observe different patterns for the different solution representations. SR-P converges quickly (within less than 100 iterations) to its best solutions, which is a good quality of a method. However, its best solutions are worse compared to those of other methods, as we saw in Table 6. Furthermore, we note that SR-C needs more iterations to get close to the best known solution, but has a steep descend. This suggests that this solution representation may perform better, if the initialization would be improved. Additionally, there is no clear difference in behaviour between the opposition based implementation and the standard implementation, when considering the same solution representation. Therefore, we chose to aggregate the GLN-PSO and GLNO-PSO results with each other and create new figures, not differentiating between obSR-1* and SR-1*.

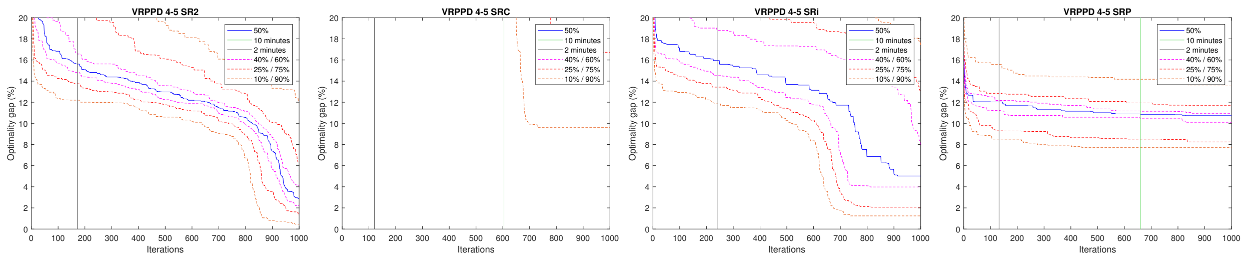
As the computational times differ per solution representation, we identify three time limits for which we chose the best solution representation.

- The 1000 iterations limit: this limit is not based on time but rather on the number of iterations. We assume in this case that the 1000 iterations represent the long run behaviour of the method, which is a strong assumption.
- The 2 minutes limit: this limit is representative for situations where time is of the essence. In this case, it is unlikely that one is able to perform a PSO method 10 times.
- The 10 minutes limit: this limit is representative for situations where there is a reasonable amount of time to perform the PSO algorithm, but not an unlimited amount of time.

When determining the best solution representation for each group, we have certain characteristics that we prefer in the behaviour of the methods.

- We aim for small optimality gaps;
- The quantile points should be relatively close to each other, as this indicates stable performance over the various runs;
- Good performance for the 10% quantile. This criterion is most important when considering the 1000 iterations and the 10 minutes limit, as for these time limits it is reasonably assumed that more runs can be performed;
- Good performance for the 90% quantile. This criterion is most important when considering the 2 minutes limit, because for these limits we reasonably assume that there is not enough time to perform multiple runs.
- Stable performance in the iterations relatively close before the time limit expires, as we could otherwise have based the decision on coincidence. This criterion is most relevant when considering the 2 and the 10 minutes limits.

We include the behavioural figures for each subgroup of VRPPD in Appendix E-J and for each subgroup of VRPSPD in Appendices K-P. In these figures a vertical lines represent the time limits. As an example, we present the behavioural plots of the SDU subgroup of the VRPPD for the different PSO methods in Figure 9.



(a) SR-2 (b) SR-C (c) SR-1* (d) SR-P

Figure 9: Development of optimality gap during 1000 iterations, where all runs of the instances 4-5 of CMTQ, CMTT and CMTH (SDU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

From Figure 9, we conclude that considering the

- 2 minutes limit: SR-P, as this has the best optimality gap for each of the quantiles that are displayed in the figures;
- 10 minutes limit: SR-2, as this has the best optimality gap for each of the quantiles that are displayed in the figures;
- Long-run: SR-2, as this has the best optimality gap for each of the quantiles that are displayed in the figures.

We make this analysis for each of the subgroups of the VRPPD and VRPSPD instances, and present the resulting decision tree in Figure 10 for VRPPD instances and in Figure 23 for the VRPSPD (to be found in Appendix Q).

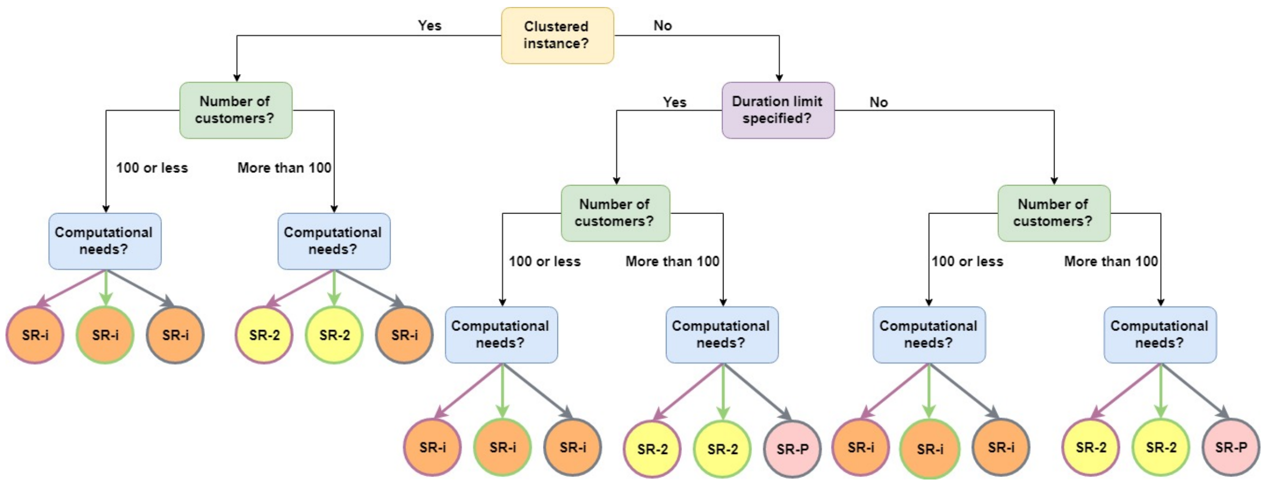


Figure 10: Decision tree with regards to the solution representation method in the VRPPD instances. Here, the left (pink aligned), middle (green aligned) and right (grey aligned) represent the best solution representation methods in the long-run, the 10 minutes time limit and the 2 minutes time limit cases respectively. All SR-i chosen cases represent SR-1*.

We clearly observe that SR-2 and SR-1* are the best performing methods. We see that the SR-C solution representation is never the best choice, making it the worst performing solution representation in any investigated scenario. Considering large customer instances, for the 2 minutes time limit, SR-P is preferred in half of the cases. As we showed in Figure 9, this is because of its fast convergence to its best computed solution. As there is no big difference with the decision three for the VRPSPD instances, we have included it in Appendix Q. We see that a clear pattern which has emerged in the VRPPD decision tree, is not present in the VRPSPD tree. For example, for the VRPPD instances, it is always preferred to use SR-1* for smaller data instances and SR-2 for the larger data instances with less restrained time limits.

6.4 Analysis of SR-P performance

We clearly observe that SR-P does not provide competitive results, which seems contradictory to the results found in Kim and Son (2012). This could be a result of the fact that this solution representation was tested on CVRP and not VRPPD or VRPSPD. However, we have seen that the problem type has not influenced the performance of the other solution representations. SR-1 and SR-2 results are competitive in this study and were competitive for the CVRP in Ai and Kachitvichyanukul (2009a). Additionally after performing SR-P on small CVRP instances, as was done by Kim and Son (2012), we see that SR-P does not yield competitive results either. Thus, it does not seem plausible that the differences in performance are primarily explained by the problem type.

As noted in Section 3.4, the SR-P was originally implemented using standard PSO instead of GLN-PSO, and implements an extra local optimization method. As we have suggested when analyzing the results in Section 6.1, PSO parameters and social learning structures can influence the performance of a solution representation.

Furthermore, the extra improvement heuristic, OR-Opt, that is used in the implementation by Kim and Son (2012), may influence the performance of SR-P. However, Babin, Deneault, and Laporte (2007) show that for arbitrary symmetric traveling salesman problems, which is the same as one VRP route, OR-Opt performs worse compared to 2-opt, when they are both implemented individually. Contrary to Babin et al. (2007), we implement several local improvement methods among which OR-Opt.

In conclusion, we think that the PSO framework, PSO parameters and the lack of OR-Opt may have contributed to the non-competitive performance of SR-P.

7 Conclusion

In this study, we have analyzed the performance of four existing solution representations on VRPPD and VRPSPD instances. Additionally, we presented a customer radius based solution representation, which we also tested on the VRPPD and VRPSPD instances. There exists an important trade-off between solution quality and computational time, when chosing the best PSO method. Based on the behaviour of the methods within one thousand iterations, we were able to select the best performing method considering various computational needs. In general we conclude that SR-2 and SR-1* present the best performances, taking into account computational times. Furthermore, we do not observe a significant difference in performance and behaviour when implementing the solution representations in opposition based PSO. However, we place several limitations to our research.

In the first place, we choose to use the same PSO parameters in our study. However, it may be possible that, when selecting the best parameters for each PSO method, the results would be different. Research into optimal parameter selection should be done for each of these methods in order to make a better comparison of the methods.

Furthermore, we observed that the SR-P method does not perform as presented in earlier research. SR-P converges to its best solution within less than 100 iterations, which is unlike the other solution representations that we studied. This makes it an attractive method. Further research could be focussed on comparing the solution representations, considering normal PSO and implementing OR-Opt, which ensured better performance, as presented in Kim and Son (2012).

At last, we have presented a new solution representation SR-C. This method does not perform well, especially for large customer instances. Further research, could focus on improving this solution representation, especially the initialization procedure of the decoding algorithm.

References

Ai, T. J., & Kachitvichyanukul, V. (2009a). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, *56*(1), 380–387.

Ai, T. J., & Kachitvichyanukul, V. (2009b). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, *36*(5), 1693–1702.

Babin, G., Deneault, S., & Laporte, G. (2007). Improvements to the or-opt heuristic for the symmetric travelling salesman problem. *Journal of the Operational Research Society*, *58*(3), 402–407.

Balinski, M. L., & Quandt, R. E. (1964). On an integer program for a delivery problem. *Operations Research*, *12*(2), 300–304.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, *12*(4), 568–581.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., & Vigo, D. (2007). Vehicle routing. *Handbooks in operations research and management science*, *14*, 367–428.

Dantzig, G., & Ramser, J. (1959). The truck dispatching problem. management science.. 6: 80-91.

Golden, B. L., Magnanti, T. L., & Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks*, *7*(2), 113–148.

Kachitvichyanukul, V., Purintrapiban, U., & Utayopas, P. (n.d.). A non-homogenous particle swarm optimization with multiple social structures.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ieee international conference on neural networks* (Vol. 4, pp. 1942–1948).

Kim, B.-I., & Son, S.-J. (2012). A probability matrix based particle swarm optimization for the capacitated vehicle routing problem. *Journal of Intelligent Manufacturing*, *23*(4), 1119–1126.

Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations research*, *33*(5), 1050–1073.

Nagy, G., & Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European journal of operational research*, *162*(1), 126–141.

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation*, *12*(1), 64–79.

Shi, Y., & Eberhart, R. C. (2008). Population diversity of particle swarms. In *2008 ieee congress on evolutionary computation (ieee world congress on computational intelligence)* (pp. 1063–1067).

Wang, H.-F., & Chen, Y.-Y. (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, *62*(1), 84–95.

A Table AI VRPPD

Data instance	Costs (€) AK	SR-1	Deviation	SR-1^0	Deviation
CMT1H	464	469.2	1.1	465.7	0.4
CMT2H	668	721	7.9	692.3	3.6
CMT3H	701	785.4	12.0	769.7	9.8
CMT4H	883	1057.7	19.8	1031.2	16.8
CMT5H	1044	1246.9	19.4	1191.1	14.1
CMT6H	557	567.6	1.9	558.7	0.3
CMT7H	943	1003.9	6.5	938.6	-0.5
CMT8H	899	956.3	6.4	875.5	-2.6
CMT9H	1207	-	-	1211.1	0.3
CMT10H	1499	1636.3	9.2	1509.3	0.7
CMT11H	830	864.7	4.2	846	1.9
CMT12H	635	703.5	10.8	653	2.8
CMT13H	1565	1649.3	5.4	1577.7	0.8
CMT14H	824	885.5	7.5	843	2.3
CMTH	**908.5**	**1022.2**	**12.5**	**940.2**	**3.5**
CMT1T	520	524.1	0.8	520.1	0
CMT2T	810	850.3	5	815.5	0.7
CMT3T	827	863.7	4.4	812	-1.8
CMT4T	1014	1132.3	11.7	1029.6	1.5
CMT5T	1297	1469.1	13.3	1295	-0.2
CMT6T	555	568.6	2.5	556.7	0.3
CMT7T	942	1030.8	9.4	951.6	1
CMT8T	904	946	4.6	918.1	1.6
CMT9T	1206	1201.3	-0.4	infeasible	-
CMT10T	1501	1661.6	10.7	1580.8	5.3
CMT11T	1026	1062.7	3.6	1033.3	0.7
CMT12T	792	829.5	4.7	795.1	0.4
CMT13T	1548	1621	4.7	1580.8	2.1
CMT14T	846	911.1	7.7	839.7	-0.7
CMTT	**950.3**	**1013.8**	**6.7**	**957.5**	**0.8**
CMT1Q	490	494.2	0.9	489.7	-0.1
CMT2Q	739	774.8	4.8	751.3	1.7
CMT3Q	768	797.8	3.9	752.3	-2
CMT4Q	938	1055	12.5	935.8	-0.2
CMT5Q	1174	1308.4	11.4	1200.1	2.2
CMT6Q	557	567.5	1.9	556.7	-0.1
CMT7Q	933	1005	7.7	943.8	1.2
CMT8Q	890	931.8	4.7	871.3	-2.1
CMT9Q	1214	1375.4	13.3	1260.1	3.8
CMT10Q	1509	infeasible	-	1500.8	-0.5
CMT11Q	964	1003.6	4.1	968	0.4
CMT12Q	733	837.3	14.2	752.8	2.7
CMT13Q	1570	1637	4.3	1590.6	1.3
CMT14Q	825	896.8	8.7	831.3	0.8
CMTQ	*984.9*	*1048*	*6.4*	*995*	*1*

Table 7: A comparison of VRPPD results, where the column of results (AK) represent the results as stated in Ai & Kachitvichyanakul (2009).

B Table AI VRPSPD

Data instance	Costs (€) AK	Costs (€) SR-1	Deviation (%)	Costs (€) SR-1^0	Deviation (%)
CMT1X	467	475.2	1.8	472.4	1.2
CMT2X	710	740.5	4.3	707.5	-0.4
CMT3X	738	754.5	2.2	740.3	0.3
CMT4X	912	1037	13.7	1018.6	11.7
CMT5X	1167	1315.8	12.8	1166.2	-0.1
CMT6X	557	565.2	1.5	558.7	0.3
CMT7X	919	infeasible	-	917.3	-0.2
CMT8X	896	953.1	6.4	909.2	1.5
CMT9X	1225	1354.9	10.6	1232.7	0.6
CMT10X	1520	1689.1	11.1	1518.3	-0.1
CMT11X	895	940.1	5.0	927.1	3.6
CMT12X	691	706.9	2.3	682.7	-1.2
CMT13X	1560	1646.6	5.6	1564.2	0.3
CMT14X	826	910.3	10.2	822.2	-0.5
CMTX	*934.5*	*1000.6*	*7.1*	*945.5*	*1.2*
CMT1Y	467	463.476	-0.8	461.241	-1.2
CMT2Y	710	684.5157	-3.6	662.386	-6.7
CMT3Y	740	748.1214	1.1	723.7468	-2.2
CMT4Y	913	918.6827	0.6	839.2346	-8.1
CMT5Y	1142	1177.015	3.1	1027.528	-10
CMT6Y	557	569.665	2.3	556.6791	-0.1
CMT7Y	934	infeasible	-	910.1357	-2.6
CMT8Y	902	971.8643	7.7	904.1619	0.2
CMT9Y	1230	1347.641	9.6	1241.137	0.9
CMT10Y	1485	1653.935	11.4	1545.153	4.1
CMT11Y	900	806.917	-10.3	784.9185	-12.8
CMT12Y	697	675.3429	-3.1	653.6322	-6.2
CMT13Y	1568	1654.399	5.5	1565.065	-0.2
CMT14Y	823	885.6713	7.6	839.5881	2
CMTY	*933.4*	*963.7*	*3.2*	*908.2*	*-2.7*

Table 8: Comparison AI results VRPSPD

C Table compare VRPPD

	Costs (€)						
Data instance	SR-imp	obSR-imp	SR-2	obSR-2	SR-C	obSR-C	SR-P
CMT1H	**465.1**	465.3	465.9	466.5	**465.1**	465.3	533
CMT2H	670.6	**669.7**	674	676.7	691.3	684	777.4
CMT3H	751.4	761.4	745.3	754	**735.7**	747.7	797.6
CMT4H	1003.3	953.7	942.8	**895.6**	979.6	929.8	992.7
CMT5H	1098.4	1093.3	**1047.2**	1046.4	1216.1	1245.4	1178.6
CMT6H	**558.7**	**558.7**	559.1	558.9	569.7	559.8	581.8
CMT7H	**905.1**	924.5	926.7	923.7	975.3	994.6	974
CMT8H	874	872.4	871.8	**867.6**	946.4	986.5	930.4
CMT9H	-	1211.3	1209.2	**1198.2**	1661.7	1464.4	1297.4
CMT10H	1485.7	1484.8	1470.8	**1460.4**	2219.6	1867.4	1541.9
CMT11H	**835.1**	835.3	842.1	835.5	843.2	853.6	900.7
CMT12H	639.9	**638.4**	651.6	643.7	663.9	672.4	695.8
CMT13H	1563.4	1562.9	1552.3	**1550.6**	1672.9	1650.7	1577.5
CMT14H	822.8	826	823.6	**822.2**	826	825.1	832.5
CMTH	*920.3*	*918.4*	*913*	***907.1***	*1033.3*	*996.2*	*972.2*
CMT1T	**520**	**520**	**520**	520.2	520.4	523.8	558.3
CMT2T	804	807.7	794.6	**792.7**	835.1	817	864.3
CMT3T	811.8	811.3	**808.1**	811.9	935.8	844.7	877.6
CMT4T	1021.9	1018.3	1024.5	**1013.8**	1111.4	1122.8	1103.4
CMT5T	1285.3	1301.7	1287.9	**1282.4**	1516.4	1579.3	1373.4
CMT6T	**556.7**	559	558.8	558	566.2	578.7	588.8
CMT7T	942.5	945.1	**930.1**	951.7	994	1014.3	983.9
CMT8T	886.2	882.2	**867.1**	871.9	1070.6	986	924.6
CMT9T	1230	1224.3	1212.5	**1210.3**	1658.1	1443.5	1297.5
CMT10T	1502.1	1493.2	1485.4	**1483.7**	1821.9	2207.5	1551.3
CMT11T	1019	1018.9	**1011.7**	1013.6	1021.4	1018.5	1052.9
CMT12T	795.5	**792.4**	803.7	801.2	810.7	808	817.8
CMT13T	1561.2	1570.2	1550.8	**1547.5**	1673.1	1661.6	1581
CMT14T	853.3	849.2	**832.7**	848.8	841.7	847.6	854
CMTT	*985*	*985.3*	***977.7***	*979.1*	*1098.3*	*1103.8*	*1030.6*
CMT1Q	490.5	490.5	490.8	490.5	**489.7**	**489.7**	529.4
CMT2Q	746.7	747.4	745.5	**744.1**	767.1	755.2	798.9
CMT3Q	765.4	**760.8**	762.6	762.2	779.9	810.7	825.6
CMT4Q	938.1	949.1	**937.3**	939.1	1007.7	1025.1	1018.4
CMT5Q	1236.3	1204.2	1190.1	**1184.6**	1290.2	1346.5	1277.9
CMT6Q	**558.7**	**558.7**	559.3	558.9	562.3	571.1	589
CMT7Q	936.3	924.9	**911**	912.9	999.8	1033.3	976.5
CMT8Q	876.8	871	**865.5**	869.9	914.1	969.5	917.6
CMT9Q	1225.3	1222.4	1218.3	**1198.9**	1633	1442.3	1286.2
CMT10Q	1508.7	1500.8	**1461.9**	1469.9	1814.1	2278.4	1558.6
CMT11Q	967.4	965.1	**955.1**	964.7	969.3	975.5	1001.9
CMT12Q	742.6	741.9	**732.6**	**732.6**	756.7	757.9	784.8
CMT13Q	1564.1	1560.6	**1549.8**	1556.9	1693.5	1621.8	1583.6
CMT14Q	827.5	829	**822.8**	**822**	824.1	826.8	832.3
CMTQ	*956*	*951.9*	***943***	*943.4*	*1035.8*	*1064.5*	*998.6*

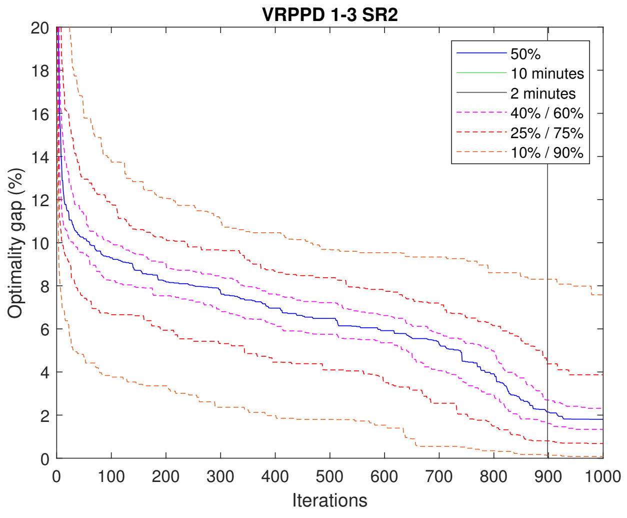
Table 9: VRPPD results for different PSO methods

D Table compare VRPSPD

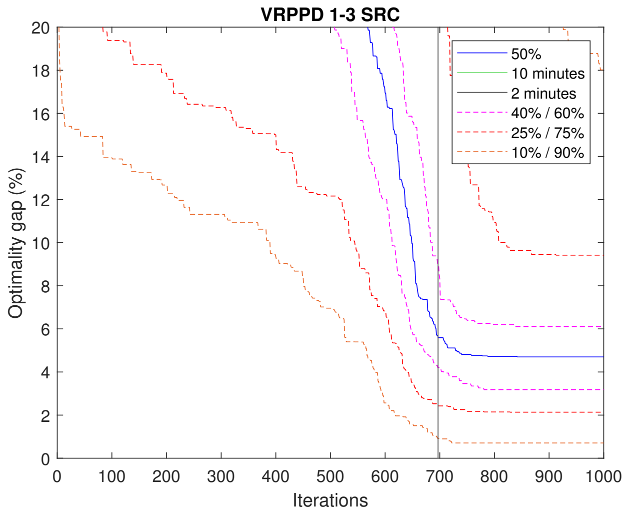
	Costs (€)						
Data instance	SR-imp	obSR-imp	SR-2	obSR-2	SR-C	obSR-C	SR-P
CMT1X	472.4	472.4	472.4	472.4	472.4	472.6	476.8
CMT2X	713.1	713.3	711.7	712.6	736.6	737.3	740.5
CMT3X	737.3	740.1	740.3	739.4	783.1	765.4	754.5
CMT4X	954.7	948.8	930.8	968.5	1018.9	985.1	1037
CMT5X	1166.4	1153.4	1115.7	1131.9	1361.1	1306.4	1315.8
CMT6X	558.7	558.7	558.9	559	559.3	568.1	565.5
CMT7X	910.5	912.9	903.6	901.8	1006.1	978.6	917.4
CMT8X	874.8	874.5	875.5	874.9	955.4	989.1	953.1
CMT9X	1220.5	1218.9	1210.6	1214.2	1611	1444.2	1354.9
CMT10X	1491.2	1491.4	1460.6	1470.1	2062.2	2213.5	1670.1
CMT11X	897.7	903.2	861.4	865.3	916.1	886.9	940.1
CMT12X	674.2	673.3	676	680.9	686.9	694.5	706.9
CMT13X	1563.5	1564.9	1550.7	1550.5	1651.9	1660.8	1646.6
CMT14X	822.3	822.8	822.2	823.4	824.1	824.4	912.2
CMTX	*932.7*	*932*	*920.7*	*926.1*	*1046.1*	*1037.6*	*999.4*
CMT1Y	461.2	461.2	461.9	462.8	462.2	461.2	520.2
CMT2Y	662.4	660.4	668.7	663	673	685.4	731.4
CMT3Y	721.1	721.5	729.7	726.9	736.6	745.3	787.6
CMT4Y	839.4	841.2	855.9	846	957.5	908	955
CMT5Y	1046.7	1038.5	1037.4	1040	1169.6	1155.8	1143.2
CMT6Y	558.7	558.7	559.3	559.1	559.3	556.7	591
CMT7Y	917.9	901.2	910.2	902.1	984.9	1001.9	967.3
CMT8Y	880.2	875.9	878.5	877.2	961.2	985.6	923
CMT9Y	1228.4	1221.8	1187.7	1217.4	1385.3	1485.2	1296.4
CMT10Y	1511.8	1518.9	1474.9	1446.7	1752.8	1868	1546.2
CMT11Y	782.3	785.4	789.4	795.5	785.3	814.3	879.6
CMT12Y	633.4	633.4	632.5	632.2	658.5	658	708.7
CMT13Y	1563.8	1563	1550	1554.9	1682.7	1670.5	1572.4
CMT14Y	825.9	835.3	822.2	823.4	824.9	825.1	834.5
CMTY	*902.4*	*901.2*	*897*	*896.2*	*971*	*987.2*	*961.2*

Table 10: Different methods applied to different problems

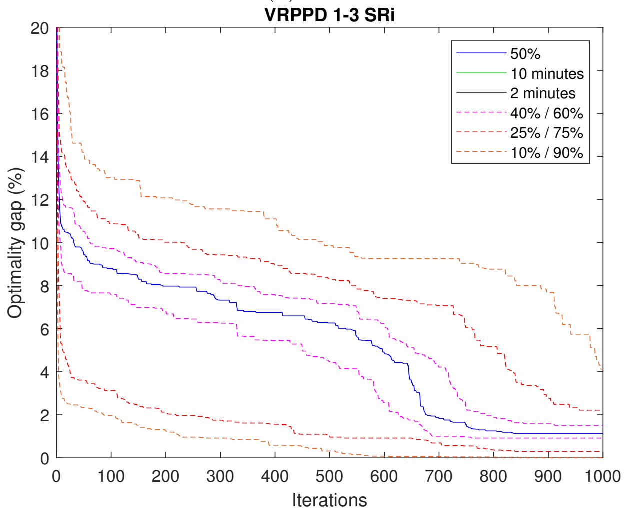
E Behavioural figures SNU VRPPD



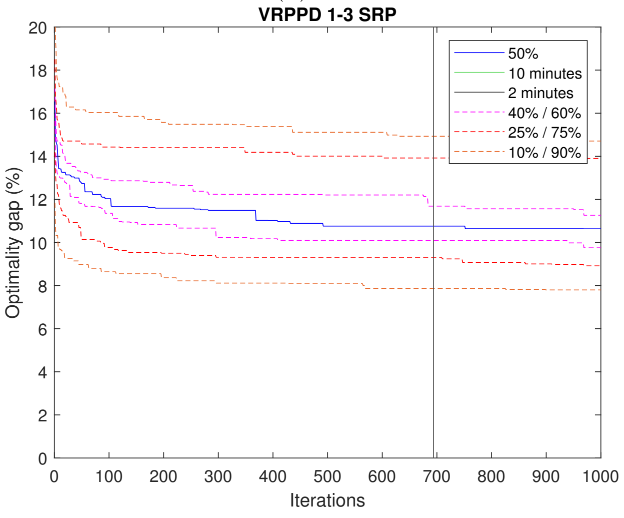
(a) SR-2



(b) SR-C



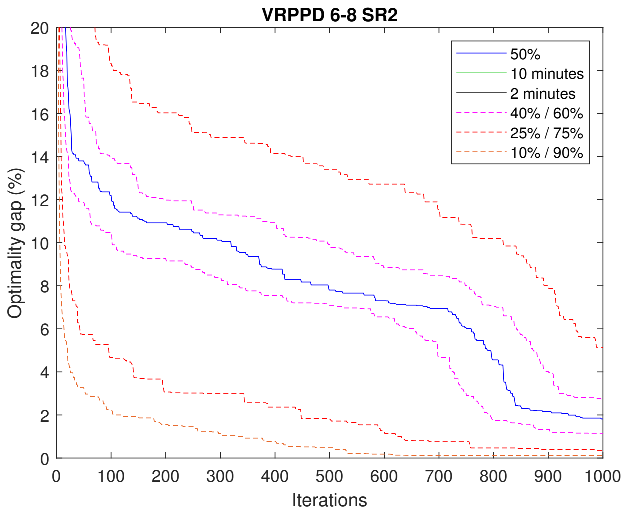
(c) SR-1*



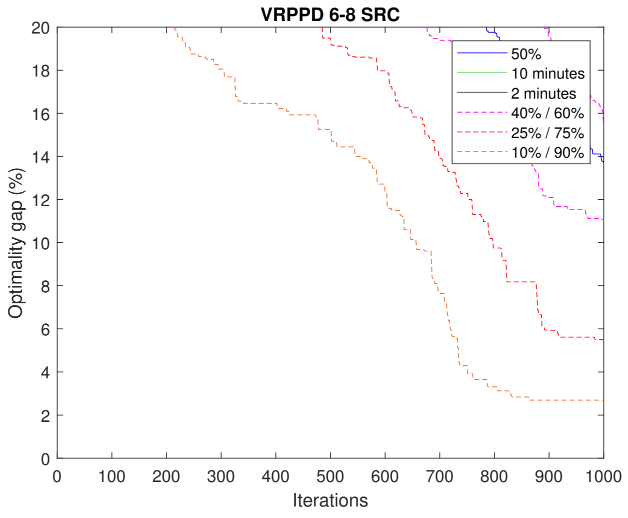
(d) SR-P

Figure 11: Development of optimality gap during 100 iterations, where all runs of the instances 1-3 of CMTQ, CMTT and CMTH (SNU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

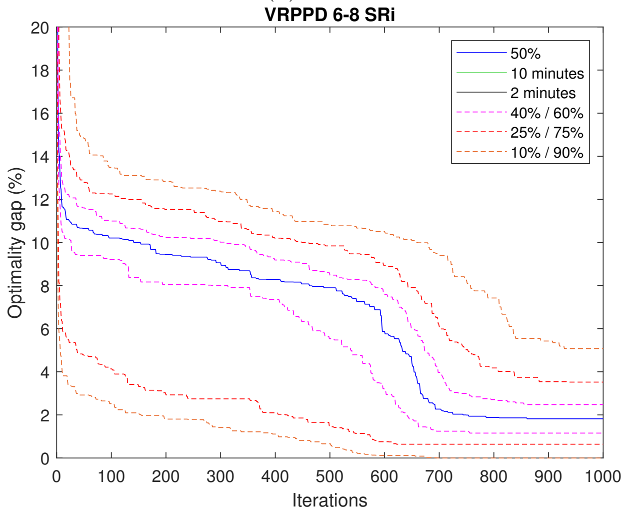
F Behavioural figures LNU VRPSPD



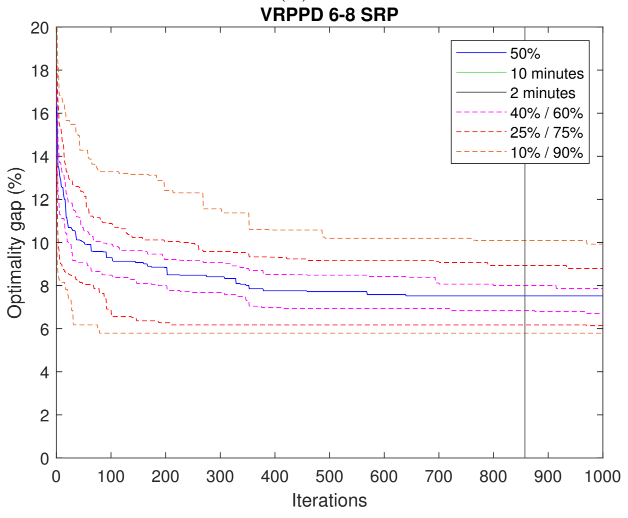
(a) SR-2



(b) SR-C



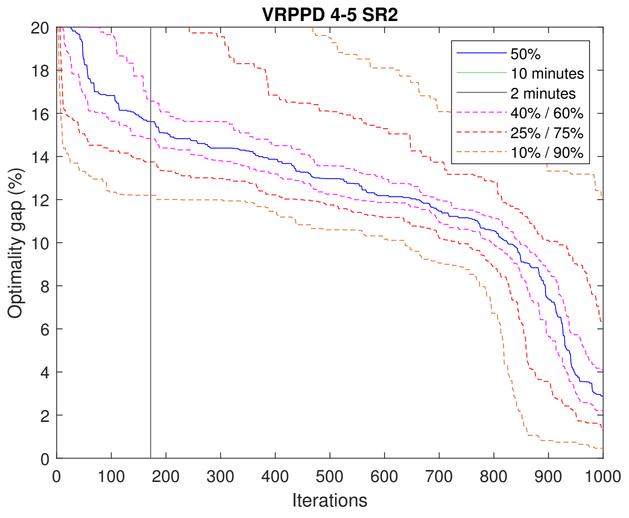
(c) SR-1*



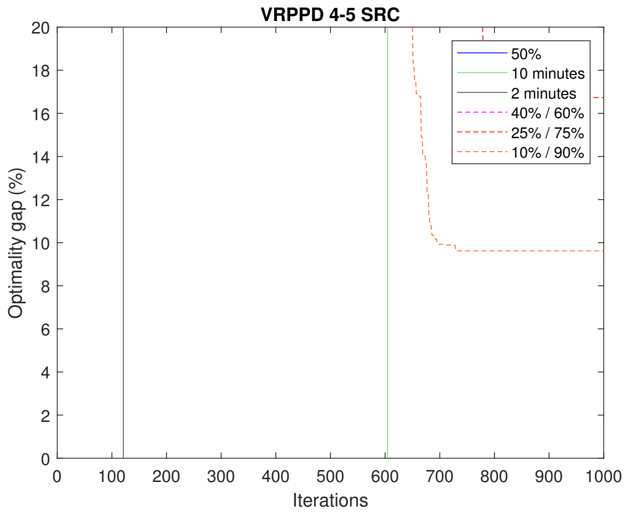
(d) SR-P

Figure 12: Development of optimality gap during 100 iterations, where all runs of the instances 6-8 of CMTQ, CMTT and CMTH (LNU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

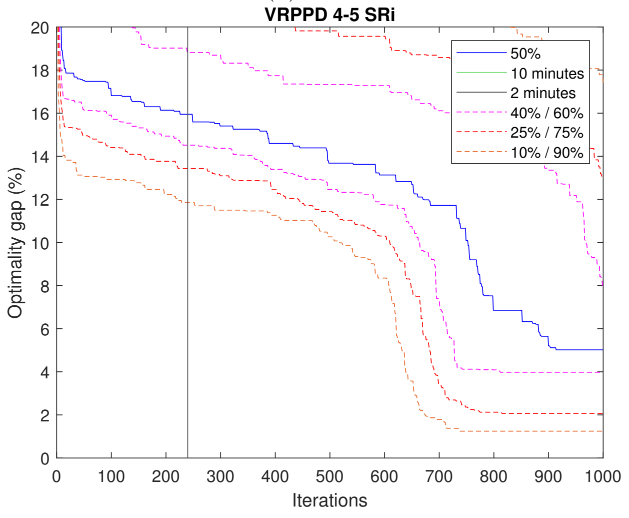
G Behavioural figures SDU VRPPD



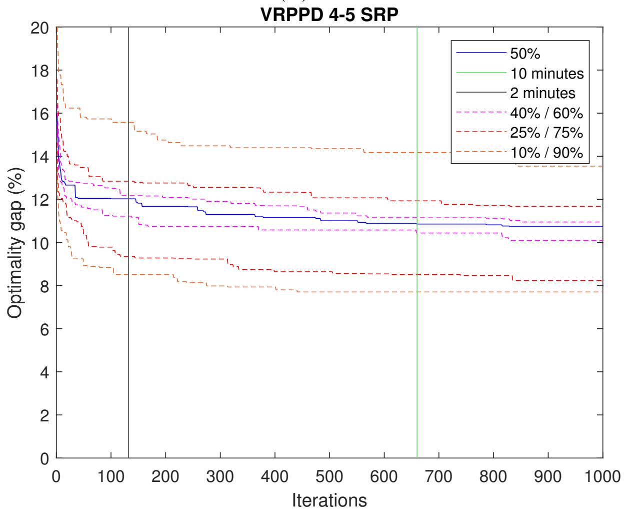
(a) SR-2



(b) SR-C



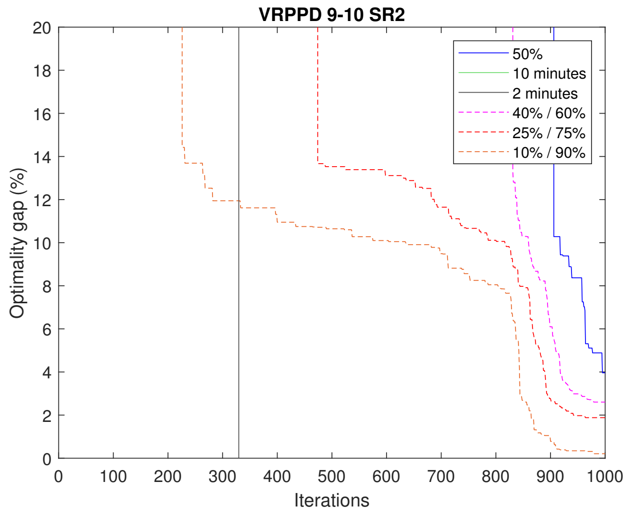
(c) SR-1*



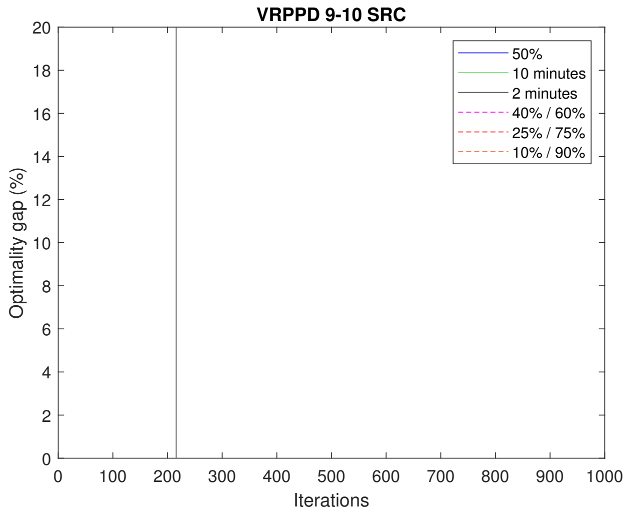
(d) SR-P

Figure 13: Development of optimality gap during 100 iterations, where all runs of the instances 4-5 of CMTQ, CMTT and CMTH (SDU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

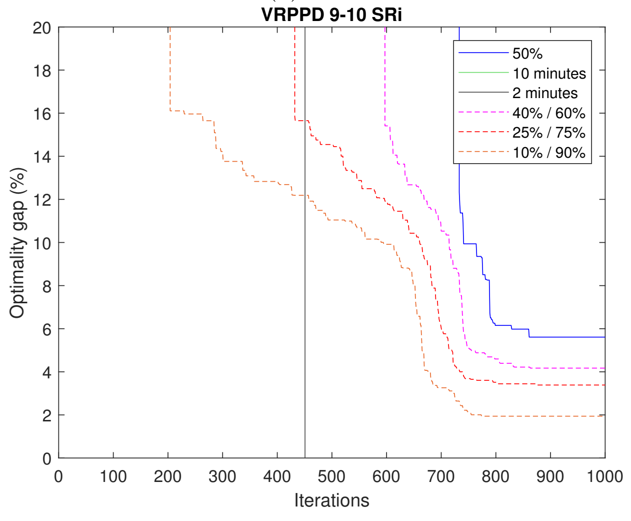
H Behavioural figures LDU VRPPD



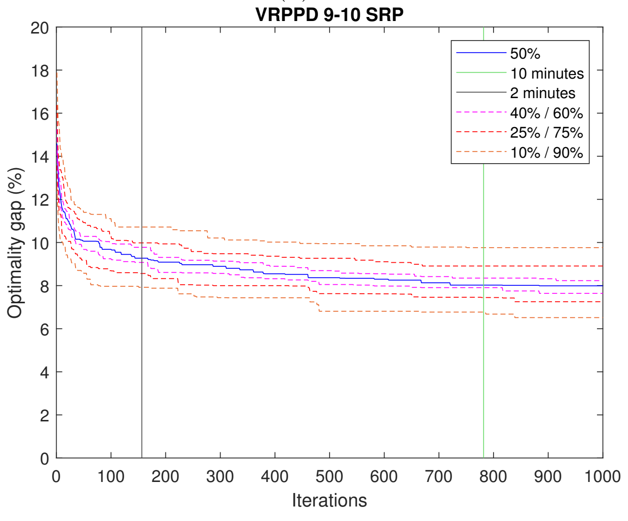
(a) SR-2



(b) SR-C



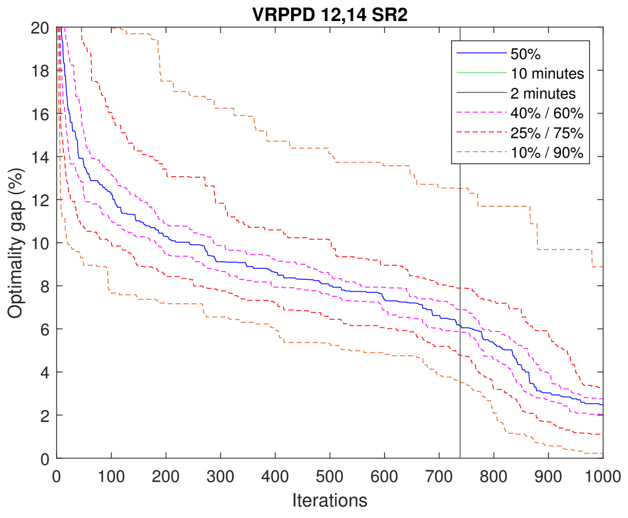
(c) SR-1*



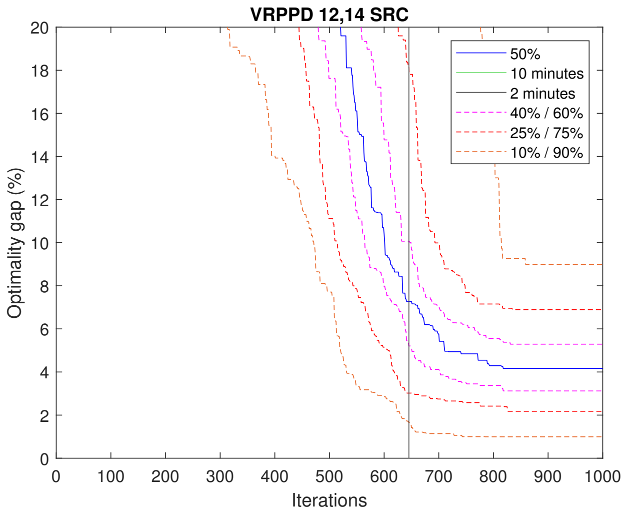
(d) SR-P

Figure 14: Development of optimality gap during 100 iterations, where all runs of the instances 9-10 of CMTQ, CMTT and CMTH (LDU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

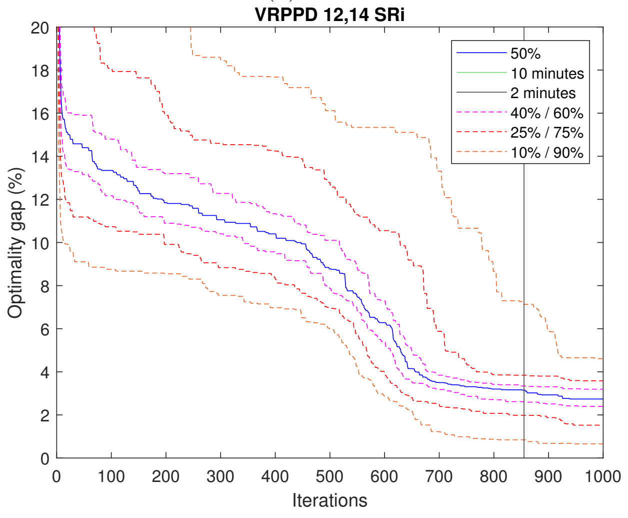
I Behavioural figures SC VRPPD



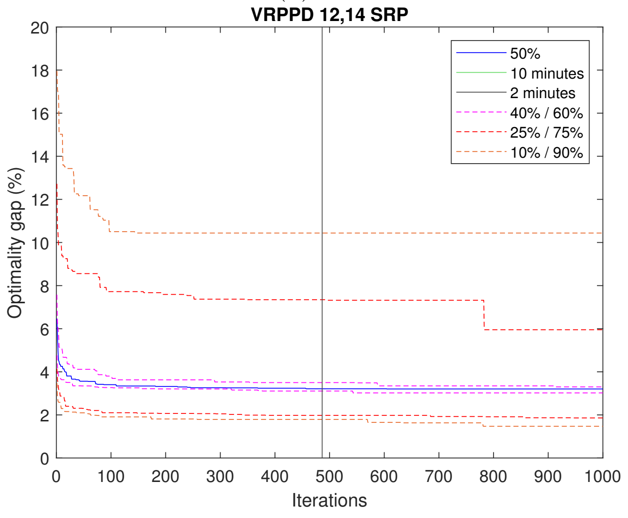
(a) SR-2



(b) SR-C



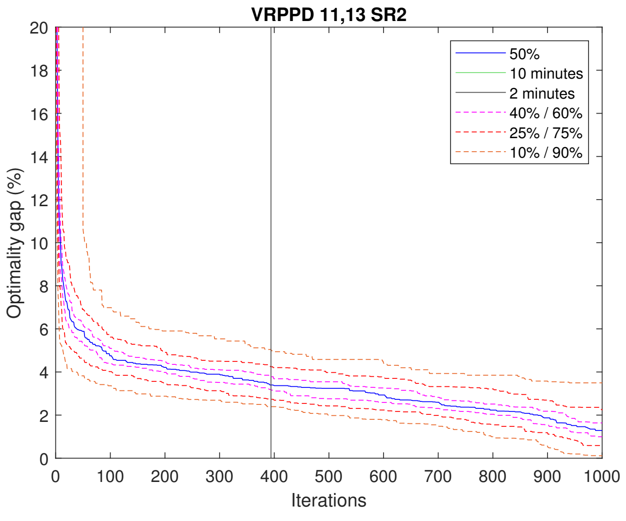
(c) SR-1*



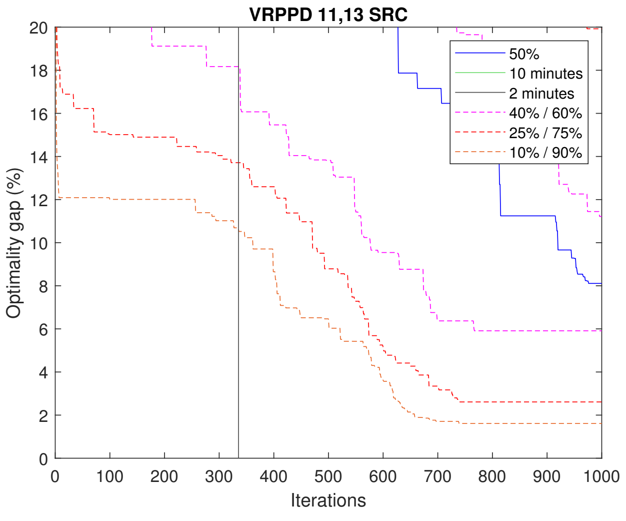
(d) SR-P

Figure 15: Development of optimality gap during 100 iterations, where all runs of the instances 12, 14 of CMTQ, CMTT and CMTH (SC) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

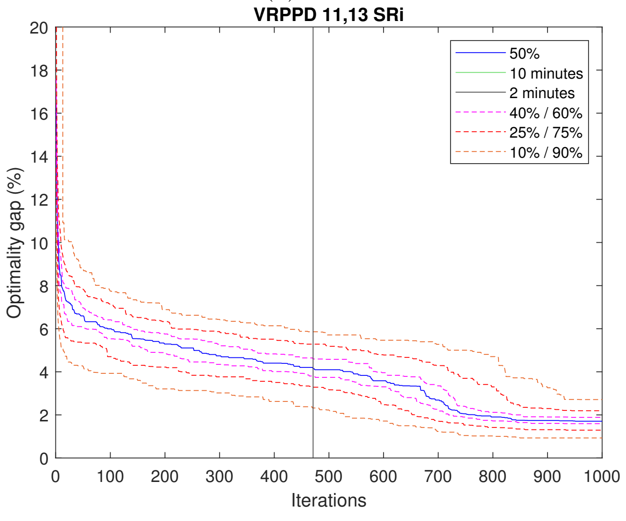
J Behavioural figures LC VRPPD



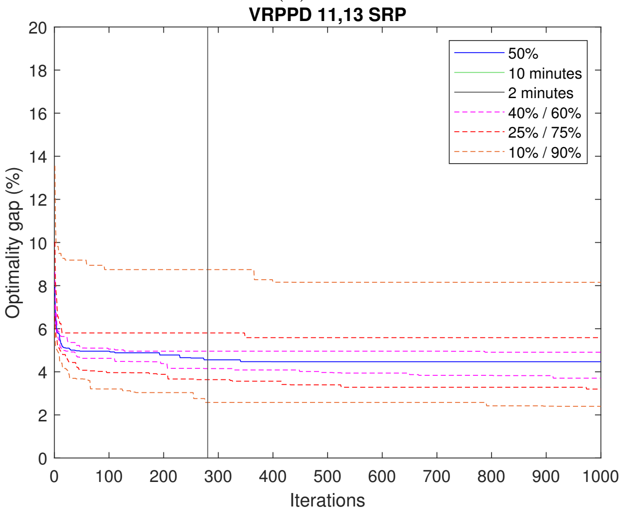
(a) SR-2



(b) SR-C



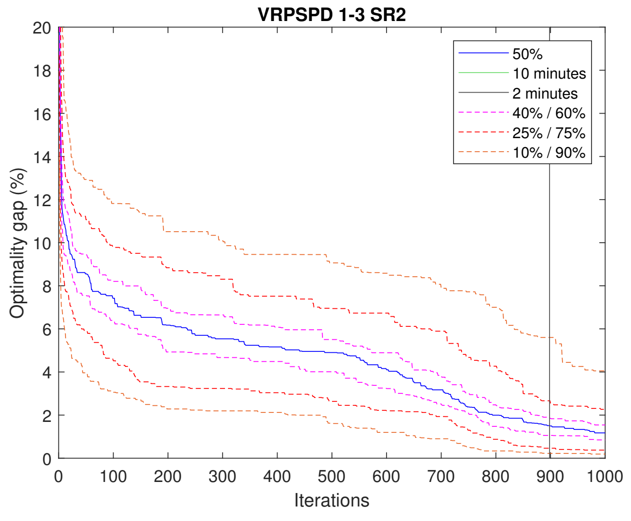
(c) SR-1*



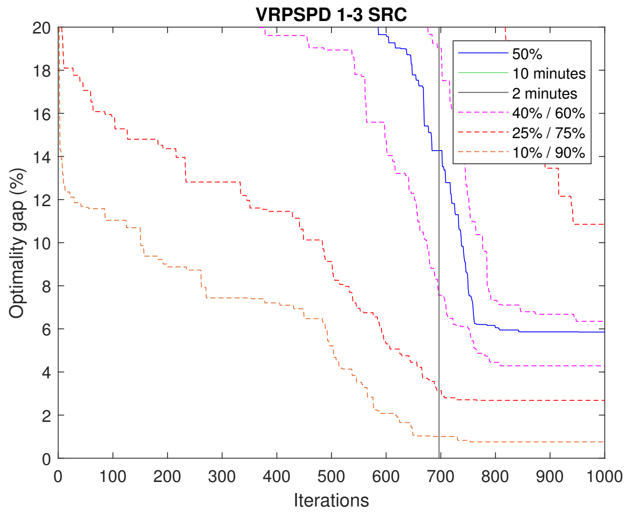
(d) SR-P

Figure 16: Development of optimality gap during 100 iterations, where all runs of the instances 11, 13 of CMTQ, CMTT and CMTH (LC) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

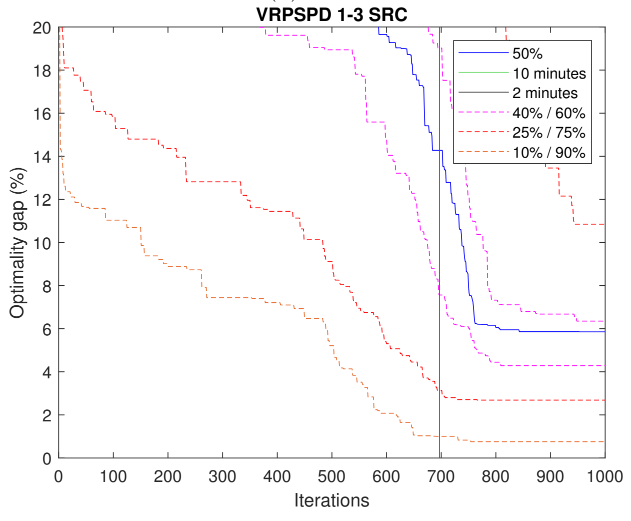
K Behavioural figures SNU VRPSPD



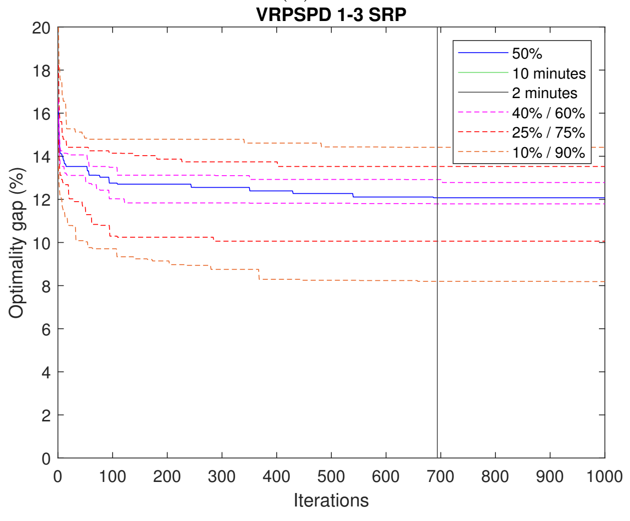
(a) SR-2



(b) SR-C



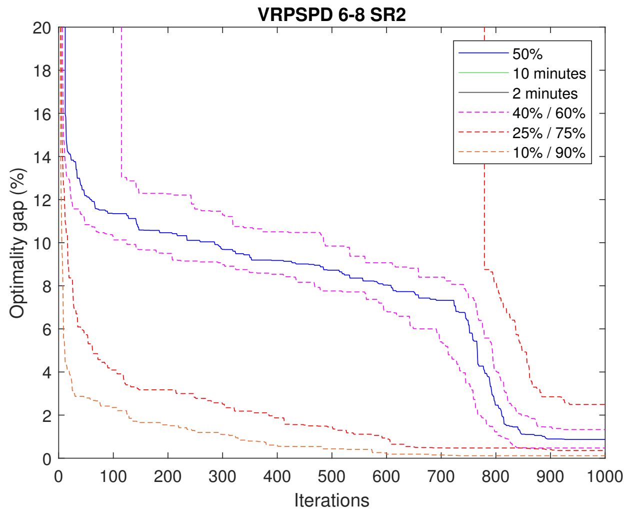
(c) SR-1*



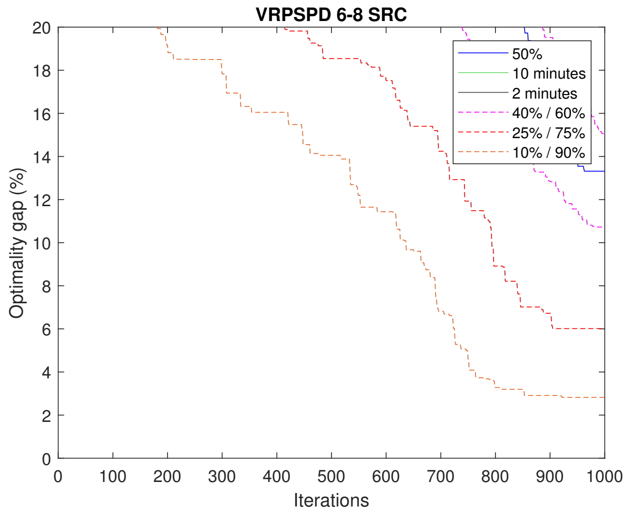
(d) SR-P

Figure 17: Development of optimality gap during 100 iterations, where all runs of the instances 1-3 of CMTX and CMTY (SNU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

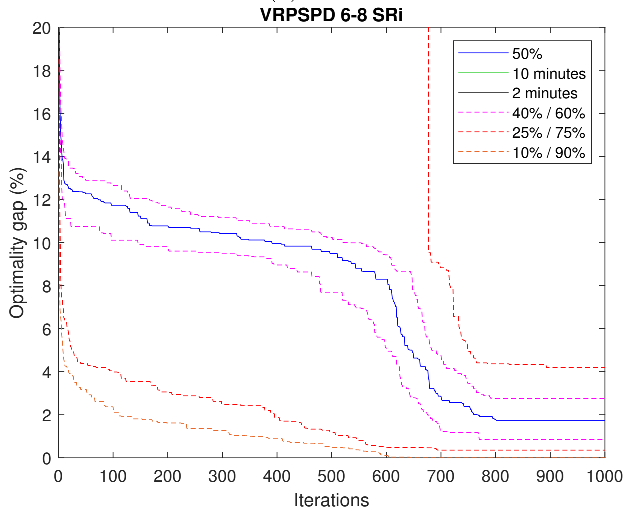
L Behavioural figures LNU VRPSPD



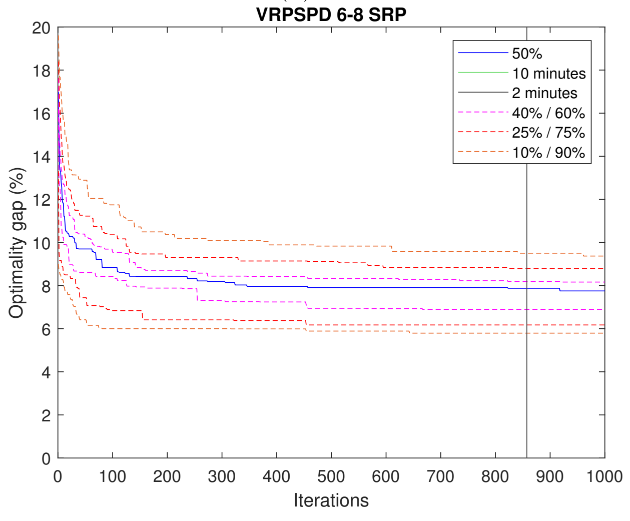
(a) SR-2



(b) SR-C



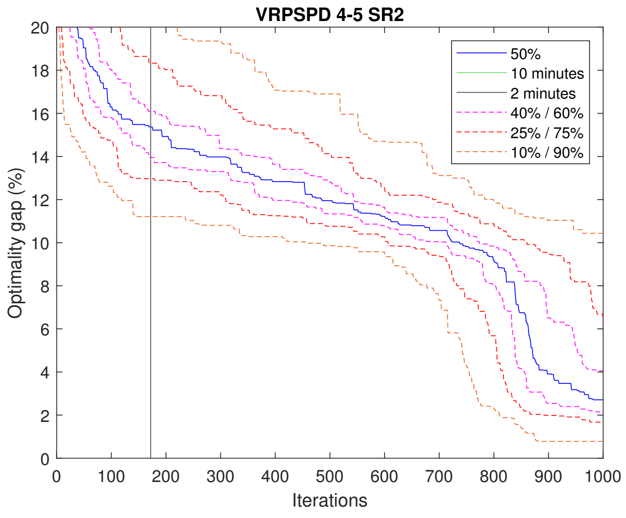
(c) SR-1*



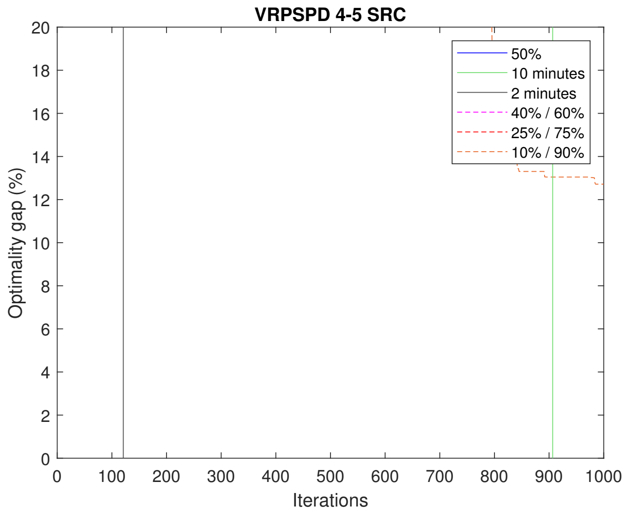
(d) SR-P

Figure 18: Development of optimality gap during 100 iterations, where all runs of the instances 6-8 of CMTX and CMTY (LNU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

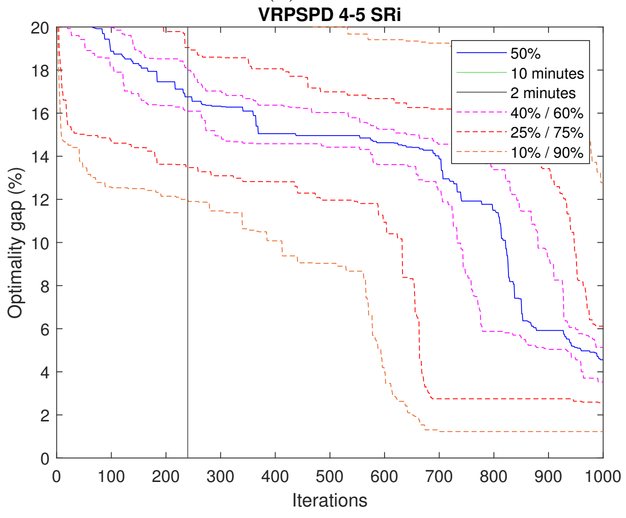
M Behavioural figures SDU VRPSPD



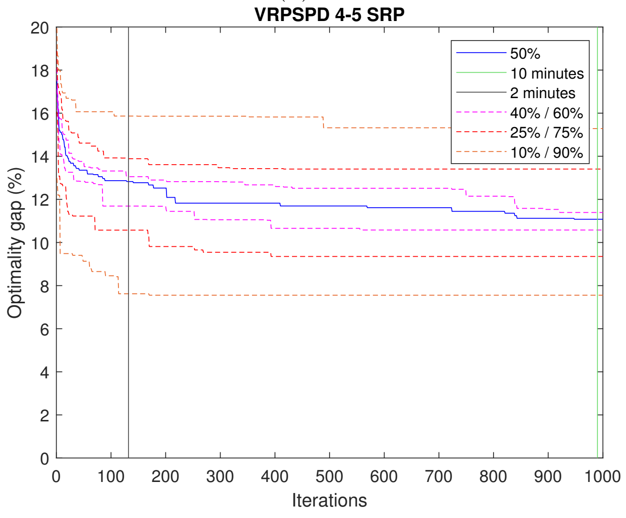
(a) SR-2



(b) SR-C



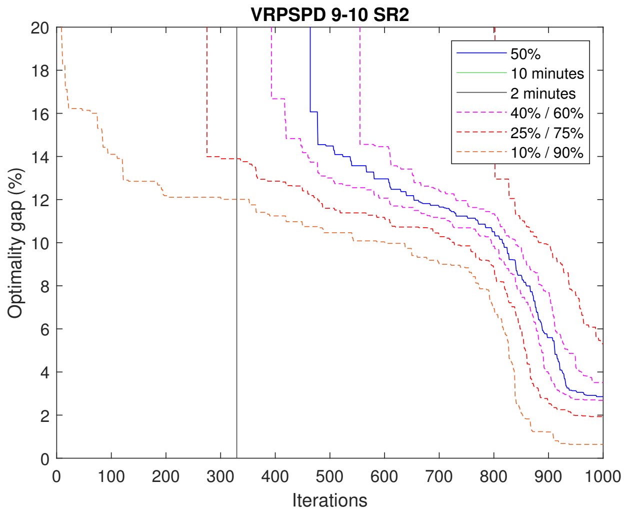
(c) SR-1*



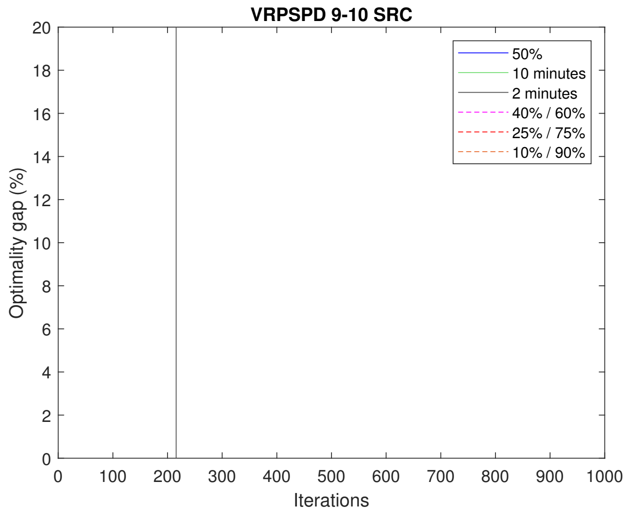
(d) SR-P

Figure 19: Development of optimality gap during 100 iterations, where all runs of the instances 4-5 of CMTX and CMTY (SDU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

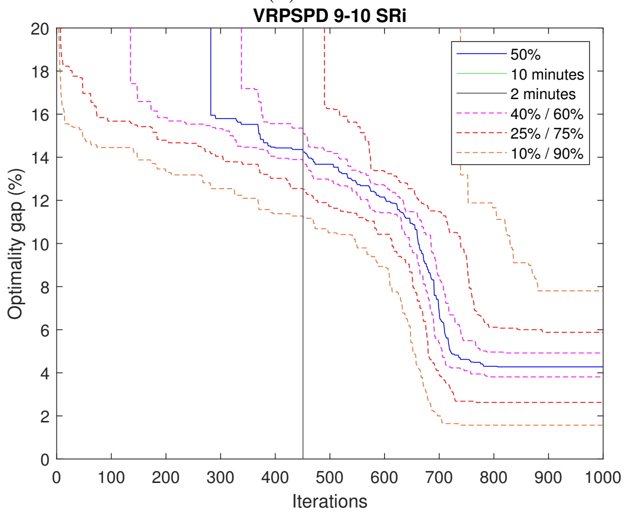
N Behavioural figures LDU VRPSPD



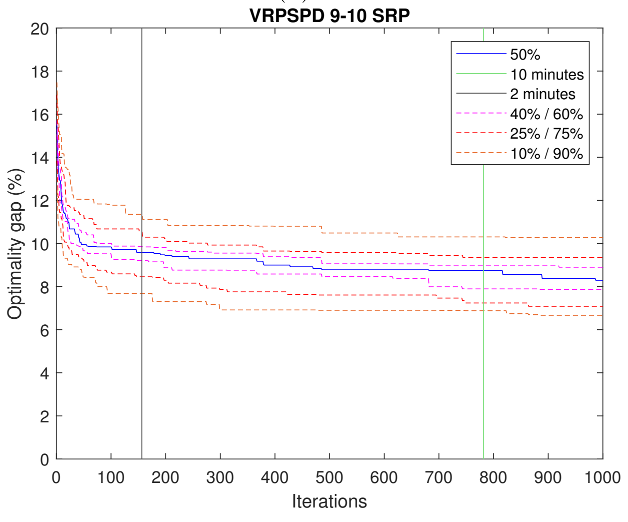
(a) SR-2



(b) SR-C



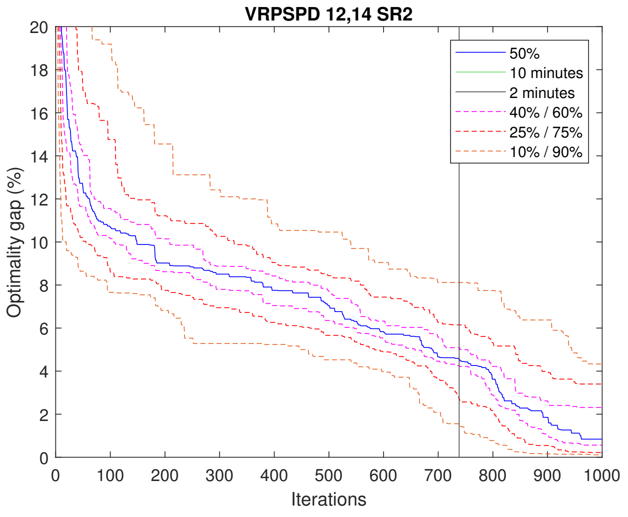
(c) SR-1*



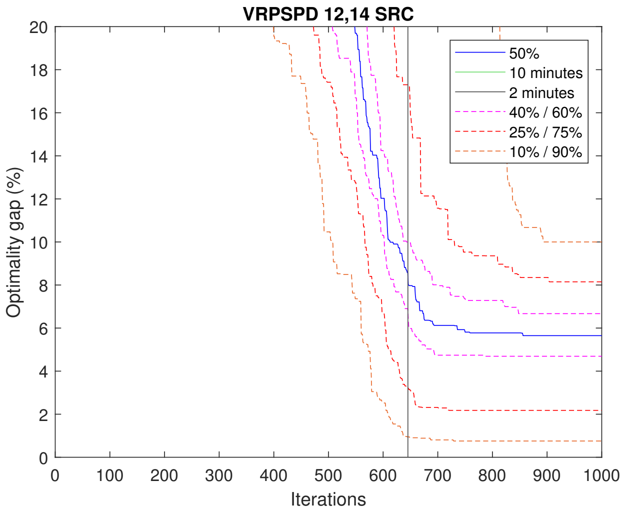
(d) SR-P

Figure 20: Development of optimality gap during 100 iterations, where all runs of the instances 9-10 of CMTX and CMTY (LDU) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

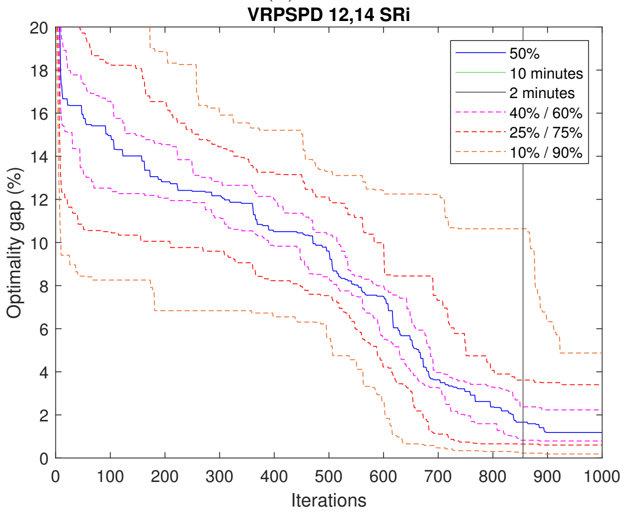
O Behavioural figures SC VRPSPD



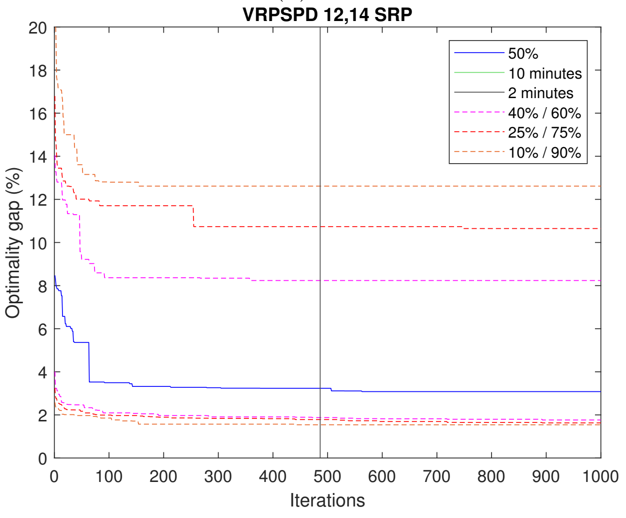
(a) SR-2



(b) SR-C



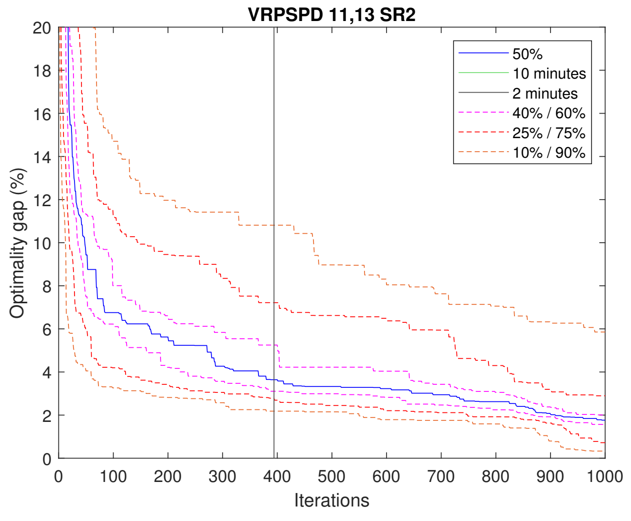
(c) SR-1*



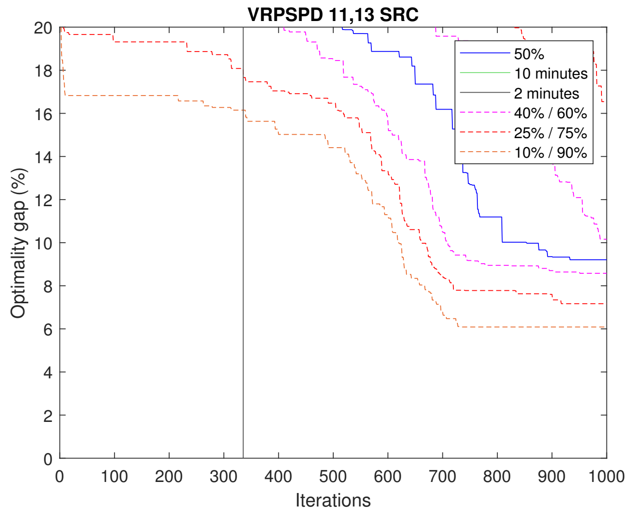
(d) SR-P

Figure 21: Development of optimality gap during 100 iterations, where all runs of the instances 12, 14 of CMTX and CMTY (SC) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

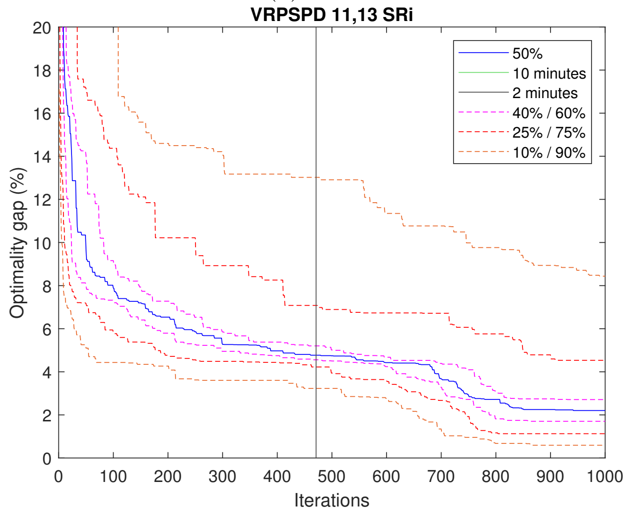
P Behavioural figures LC VRPSPD



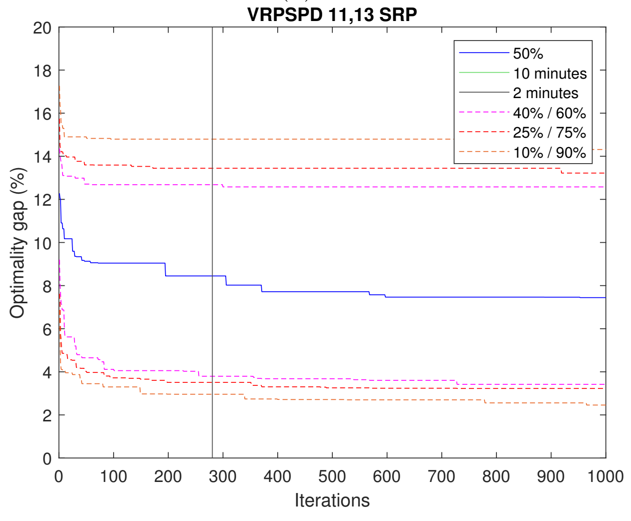
(a) SR-2



(b) SR-C



(c) SR-1*



(d) SR-P

Figure 22: Development of optimality gap during 100 iterations, where all runs of the instances 11, 13 of CMTX and CMTY (LC) are aggregated. For each method in each figure separately, the quantiles of the results regarding the optimality gap after each iteration are given. The iteration corresponding to a certain time limit is marked with a verticle line.

Q Decision tree VRPSPD

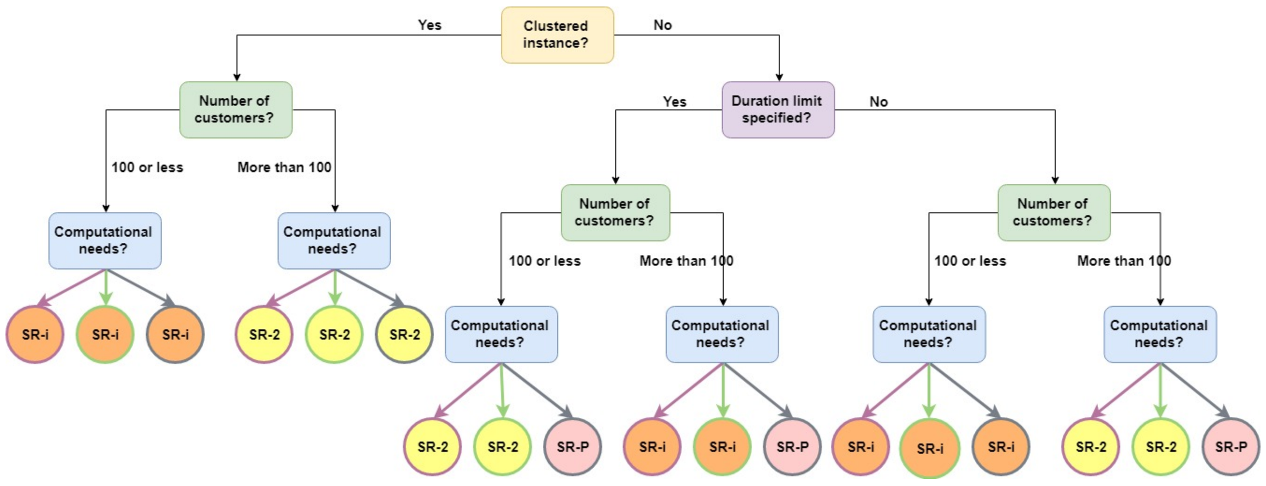


Figure 23: Decision tree with regards to the solution representation method in the VRPSPD instances. Here, the left (pink aligned), middle (green aligned) and right (grey aligned) represent the best solution representation methods in the long-run, the 10 minutes time limit and the 2 minutes time limit cases respectively. All SR-i chosen methods represent SR-1*.

R Information of included code

In this Appendix, information on the included classes is provided.

- *ExcelReader* reads an Excel file from input to an array (acquired externally based on APACHE POI package for Java users)
- *ExcelWriter* writes an ArrayList to an Excel file (acquired externally based on APACHE POI package for Java users)
- *PSOGeneral* performs GLN-PSO with SR-1
- *PSOImproved* performs GLN-PSO with SR-1*
- *PSOSR2* performs GLN-PSO with SR-2
- *SRP* performs GLN-PSO with SR-2
- *SRV* performs GLN-PSO with SR-C
- *runnen* enables us to run the PSO algorithms and reads the outcomes to an excel file