The impact of particular components of the PSO-based algorithm solving the Dynamic Vehicle Routing Problem

Michał Okulewicz[a,*], Jacek Mańdziuk[a]

[a] *Warsaw University of Technology*
Faculty of Mathematics and Information Science
Koszykowa 75, 00-662 Warsaw POLAND

Abstract

This paper presents and analyzes a Two-Phase Multi-Swarm Particle Swarm Optimizer (2MPSO) solving the Dynamic Vehicle Routing Problem (DVRP). The research presented in this paper focuses on finding a configuration of several optimization improvement techniques, dedicated to solving dynamic optimization problems, within the 2MPSO framework. Techniques, whose impact on results achieved for DVRP is analyzed, include: solving the current state of a problem with a capacitated clustering and routing heuristic algorithms, solving requests-to-vehicles assignment by the PSO algorithm, route optimization by a separate instance of the PSO algorithm, and knowledge transfer between subsequent states of the problem. The results obtained by the best chosen configuration of the 2MPSO are compared with the state-of-the-art literature results on a popular set of benchmark instances.

Our study shows that strong results achieved by 2MPSO should be attributed to three factors: generating initial solutions with a clustering heuristic, optimizing the requests-to-vehicle assignment with a metaheuristic approach, direct passing of solutions obtained in the previous stage (times step) of the problem solving procedure to the next stage. Additionally, 2MPSO outperforms the average results obtained by other algorithms presented in the literature, both in the time limited experiments, as well as those restricted by the number of fitness function evaluations.

Keywords: Dynamic Vehicle Routing Problem, Particle Swarm Optimization, Vehicle Routing Problem, Dynamic Optimization

*Corresponding author
Email addresses: M.Okulewicz@mini.pw.edu.pl (Michał Okulewicz), J.Mandziuk@mini.pw.edu.pl (Jacek Mańdziuk)

Preprint submitted to ASOC, accepted and avaialble at http://dx.doi.org/10.1016/j.asoc.2017.04.070 April 30, 2017

1. Introduction

Vehicle Routing Problems (VRP), and Dynamic Vehicle Routing Problem (DVRP) in particular, are of great theoretical and practical interest. That interest has grown since an introduction of an efficient wireless communication systems (e.g. GSM) and an accurate real-time localization services (e.g. GPS) supported by the development of the Geographical Information Systems [1]. Basic variant of the VRP has been introduced in the literature as a problem of finding a set of optimal routes for a given number of oil distributing trucks [2]. Since its introduction numerous modifications to the initial problem formulation have been proposed. One of those formulations is VRP with Dynamic Requests, most commonly referred to as a DVRP [3].

Although several types of metaheuristic methods have been applied to solve DVRP and different methods of applying Particle Swarm Optimization (PSO) to various types of the VRP have been studied, little has been done to assess the impact of various high-level (i.e. independent of the optimization algorithm) components of these optimization methods on obtained results. Some research regarding these aspects has been conducted by Mavrovouniotis and Yang [4], within the domain of Periodic VRP, concerning the composition of the initial population. Within the domain of DVRP, Khouadjia et al. [5] tested the relevance of parallelism and particular adaptation methods applied to candidate solutions found in the previous stages of the problem solving procedure.

Most of metaheuristic approaches applied to DVRP, which are reviewed in this paper, follow a hybrid optimization pattern consisting of the following modules: a metaheuristic as a main optimization engine, an additional heuristic used as a local search operator, and a solution migration scheme. In such a hybrid approach, an optimization process consists of the following steps: a population initialization (or its adaptation in the case of dynamic problems), optimization by the metaheuristic engine, its further improvement with a heuristic algorithm, and (optionally) repairement of unfeasible solutions.

The Two-Phase Multi-Swarm Particle Swarm Optimization (2MPSO) algorithm [6], developed by the authors, also follows the above-mentioned pattern. Therefore, an analysis of results obtained by various configurations of the 2MPSO can serve as a basis for studying efficiency of particular optimization techniques and solution migrations schemes. In the case of 2MPSO applied to DVRP those techniques include using a capacitated clustering algorithm for obtaining approximate solutions, optimizing requests-to-vehicles assignment in a continuous search space, creating an approximate routes from the clustered requests with a 2–OPT algorithm, fine tuning of the routes. Solution migration schemes include direct passing of the previous step solution to the next step, and following the location of tentatively assigned requests solution.

The rest of the paper is organized as follows. Section 2 gives a formal definition of the DVRP and reviews application of various metaheuristic methods. Section 3 introduces PSO algorithm, which is used as a main optimization engine in our 2MPSO method, and reviews several approaches to applying PSO to solving VRP. Section 4 describes the 2MPSO method proposed by the authors.

Section 5 presents experiments on various 2MPSO configurations and comparison of 2MPSO outcomes with literature results. The last section concludes the paper.

2. Dynamic Vehicle Routing Problem

In general, particular VRP instance is specified [2] by the properties of the following collections of objects:

- a fleet V of n vehicles,
- a series C of m clients (requests) to be served, and
- a set D of k depots from which vehicles may start their routes.

The goal of the VRP is to find an assignment of clients to vehicles and the order in which the clients' locations are to be visited by those vehicles.

Due to the fact, that various VRP models include different sets of properties and constraints, this section defines the objects in sets V, D, C and the problem constraints for the DVRP model discussed in this paper.

The fleet V is homogeneous, i.e. vehicles have identical capacity $cap \in \mathbb{R}$ and the same $speed$[1] $\in \mathbb{R}$. Vehicles are stationed in one of the k depots[2]. Each depot $d_j \in D, j = 1, \ldots, k$ has assigned

- a certain location $l_j \in \mathbb{R}^2$ and
- working hours (t_{start_j}, t_{end_j}), where $0 \leq t_{start_j} < t_{end_j}$.

For the sake of simplicity, we additionally define two global auxiliary variables (constraints): $t_{start} := \min\limits_{j \in 1,\ldots,k} t_{start_j}$ and $t_{end} := \max\limits_{j \in 1,\ldots,k} t_{end_j}$, which are not part of the standard definition.

Each client $c_l \in C$ $(l = k+1, \ldots, k+m)$, has a given:

- location $l_l \in \mathbb{R}^2$,
- time $t_l \in \mathbb{R}$, which is a point in time when their request becomes available $(t_{start} \leq t_l \leq t_{end})$,
- unload time $u_l \in \mathbb{R}$, which is the time required to unload the cargo,
- size $s_l \in \mathbb{R}$ - which is the size of the request $(s_l \leq cap)$.

[1] In all benchmarks used in this paper *speed* is defined as one distance unit per one time unit.

[2] In the most common benchmarks used in the literature, likewise in this paper, it is assumed that $k = 1$.

A travel distance $\rho(i,j)$ is the Euclidean distance between l_i and l_j in $\mathbb{R}^2$, where $i,j = 1,\ldots,k+m$.

For each vehicle v_i, $r_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,m_i})$ is a sequence of m_i indices of requests and depots assigned to be visited by the ith vehicle. Therefore, r_i defines the route of the ith vehicle. Please note, that the first and the last elements always denote depots (the initial one and the final one, respectively). The $arv_{r_{i,j}}$ is the time of arrival to the jth location on the route of the ith vehicle. $arv_{r_{i,j}}$ is induced by the permutation r_i, the time when requests become available - see eqs. (2) and (3) and the time $arv_{r_{i,1}}$ on which ith vehicle leaves the depot.

As previously stated, the goal is to serve all the clients (requests), according to their defined constraints, with minimal total cost (travel distance) within the time constraints imposed by the working hours of the depots.

In other words, the goal of the algorithm is to find such a set $R = \{r_1^*, r_2^*, \ldots, r_n^*\}$ of permutations of requests and depots that minimizes the following cost function:

$$COST(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{m_i} \rho(r_{i,j-1}, r_{i,j}) \tag{1}$$

under the following constraints (2) - (6).

Vehicle $v_i, i = 1,2,\ldots,n$ cannot arrive at location $l_{r_{i,j}}$ until the time required for traveling from the last visited location $l_{r_{i,j-1}}$ (after receiving an information about the new request) is completed:

$$\forall_{i\in\{1,2,\ldots n\}} \forall_{j\in\{2,3\ldots m_i\}} \quad arv_{r_{i,j}} \geq t_{r_{i,j}} + \rho(r_{i,j-1}, r_{i,j}) \tag{2}$$

Please recall that for $j = 2$, $l_{r_{i,j-1}}$ denotes the location of the starting depot.

Vehicle v_i cannot arrive at location $l_{r_{i,j}}$ before serving the request $c_{r_{i,j-1}}$ and traveling to the next location:

$$\begin{gathered} \forall_{i\in\{1,2,\ldots n\}} \forall_{j\in\{2,3\ldots m_i\}} \quad arv_{r_{i,j}} \\ \geq arv_{r_{i,j-1}} + u_{r_{i,j-1}} + \rho(r_{i,j-1}, r_{i,j}) \end{gathered} \tag{3}$$

All vehicles must return to the depot before its closing and cannot leave the depot before its opening:

$$\begin{gathered} \forall_{i\in\{1,2,\ldots n\}} \quad arv_{r_{i,1}} \geq t_{start_{r_{i,1}}} \\ \forall_{i\in\{1,2,\ldots n\}} \quad arv_{r_{i,m_i}} \leq t_{end_{r_{i,m_i}}} \end{gathered} \tag{4}$$

Recall that index r_{i,m_i} (the last index in route r_i) denotes the closing depot for vehicle i.

A sum of requests' sizes between consecutive visits to the depots must not exceed vehicle's capacity:

$$\forall_{i\in\{1,2,\ldots n\}}\forall_{j_1<j_2\in\{1,2\ldots m_i\}} \quad (r_{i,j_1} \text{ and } \mathrm{r_{i,j_2}} \text{ are two subsequent visits to the depots in route } r_i) \;\Rightarrow\; (\sum_{j=j_1+1}^{j_2-1} s_{r_{i,j}} \leq cap) \tag{5}$$

Each client must be assigned to exactly one vehicle:

$$\forall_{j\in\{1+k,2+k,\ldots m+k\}}\exists!_{i\in\{1,2,\ldots n\}} \;\; j \in r_i \tag{6}$$

2.1. Dynamic Vehicle Routing Problem solving framework

On a general note there are two major approaches to solving dynamic optimization problems, dynamic transportation problems in particular. In the first one the optimization algorithm is run continuously, adapting to the changes in the environment [7]. In the second one, time is divided into discrete slices and the algorithm is run once per time slice, usually at its origin, and the problem instance is "frozen" for the rest of the time slice period. In effect, any potential changes introduced during the current time slot are handled in the next run of the algorithm, which is scheduled for the subsequent time slice period.

In this study the latter approach, which in the context of DVRP was proposed by Kilby et al. [8], is adopted.

In a typical approach to solving DVRP, regardless of the particular optimization method used, one utilizes a vehicles' dispatcher (event scheduler) module, which is responsible for communication issues. In particular, the event scheduler collects information about new clients' requests, generates the current problem instance and sends it to the optimization module and, afterwards, uses the solution found to commit vehicles. Such a DVRP processing scheme is depicted in Fig. 1, while a technical description of such information technology system could be found in [9].

The processing of the DVRP is controlled by the following parameters, maintained by an event scheduler, which in some sense define the "degree of dynamism" of a given problem instance:

- T_{co} - *cut-off* time,
- n_{ts} - number of *time slices*,
- T_{ac} - *advanced commitment* time.

The *cut-off* time (T_{co}), in real business situations, could be interpreted as a time threshold for not accepting any new requests that arrive after T_{co} and treating them as the next-day's requests, available at the beginning of the next working day. In a one-day simulation horizon considered in this paper, likewise in the referenced works [5, 6, 10–17], the requests that arrive after the T_{co} are treated as being known at the beginning of the *current* day, i.e. they actually

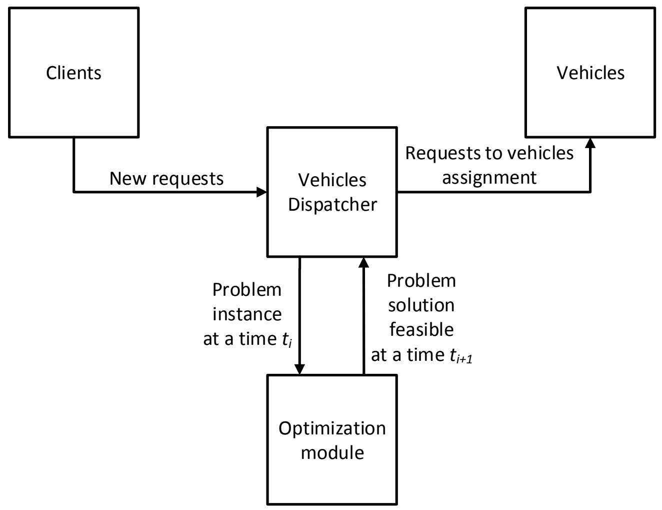


Figure 1: High-level diagram of DVRP solving framework.

compose the initial problem instance. In all tests, for the sake of comparability with the previous results, $T_{co} = 0.5$ was set, so as to make this choice consistent with the above-cited works.

The number of *time slices* (n_{ts}) decides how often the dispatcher sends a new version of the problem to the optimization module. Kilby et al. [8] set this value to 50, while Montemanni et al.[10] proposed 25 (adopted as a standard value in other subsequent approaches), claiming the optimal trade-off between the quality of solutions and computation time. In the case of our method we observed that it is beneficial to set n_{ts} to 40. Generally speaking, dividing the day into greater number of time slices allows optimization module to react faster to the newly-arrived requests since it is informed sooner about the introduced changes. On the other hand, with the fitness function evaluations (FFE) or computations time budget fixed the chances for optimizing the solution within each time slice decrease proportionally. For the sake of comparability with our previous work [6] and with MEMSO algorithm [5] we have conducted experiment with the total number of FFEs equal bound by 10^6, while in order to compute results comparable with the GA [13], EH [15] and ACO [16] we have conducted experiments with the total computations time limited to 75 seconds per benchmark.

The *advanced commitment* time (T_{ac}) parameter is a safety buffer, which shifts the latest possible moment in which a current part of the route is ultimately approved and "frozen", i.e. the vehicle is dispatched to serve the respective requests. In other words, any vehicles expected to return to depot within the last time slice before its closing time minus T_{ac} are considered close to fulfilling time constraint defined by eq. (4), and need to be dispatched:

$$V_{tbd} = \{v_i : arv_{r_{i,m_i}} \geq t_{end_{r_{i,m_i}}} - (T_{ac} + \frac{1}{n_{ts}}) \left(t_{end_{r_{i,m_i}}} - t_{start r_{i,1}}\right)\} \tag{7}$$

Requests scheduled to be served by a vehicle from a V_{tbd} set within the closest time slice are treated as ultimately approved and cannot be rescheduled to another vehicle. Please note, that new requests can be added to the dispatched vehicle and tentatively assigned requests can still be rescheduled to another vehicle.

We have observed that appropriate choice of T_{ac} allows greater flexibility in assigning requests to vehicles in the phase of a day, just before the T_{co}, when appropriate handling of potential arrival of a large request is a critical issue.

2.2. Metaheuristics applied to Dynamic Vehicle Routing Problem

As observed in [18] the methods of solving VRP problems might be categorized into seven types, depending on the method of assigning requests to vehicles and construct routes. The algorithms discussed in this paper fall into three of those categories: *cluster-first route-second*, *route-first cluster-second* and *improvement and exchange*.

In the *cluster-first route-second* category, the requests are first assigned to vehicles and subsequently ordered (separately within each vehicle). In the *route-first cluster-second* approach category, the requests are first ordered in a giant TSP tour and subsequently divisioned among the vehicles. In the *improvement and exchange* approach category, the initial candidate solutions are created as a feasible ones and subsequently improved by various search operators. Classification of the reviewed methods applied to DVRP into those categories is presented in Tab. 1.

The first metaheuristic approach applied to DVRP has been an Ant Colony System (ACS) [10]. In that approach a direct modification of ACS for the Traveling Salesman Problem (TSP) has been applied. Each ant traversed a whole graph of requests returning to the depot if needed. The highest levels of pheromone have been applied on the routes forming a candidate solution with the shortest total routes' length.

The subsequent approaches utilized Genetic Algorithm (GA) and Tabu Search (TS) [13]. In that approach a problem solution has also been coded as a giant TSP tour, but without the visits to a depot. In that GA and TS requests on the tour are divided among the vehicles by a greedy rule (i.e. are assigned to the same vehicle as long as the capacity (eq. (5)) and time (eq. 4)) constrains are satisfied).

Methods based on that initial research included: an adaptive heuristic building Evolutionary Hyperheuristic (EH) [15, 19, 20], a Memetic Algorithm (MA) consisting of GA with a local search based on adaptive heuristic operators sequences [17], encoding depots within the giant tour encoding and changing the

Table 1: Summary of different metaheuristics applied to DVRP.

Authors	(Year)	Category	Search space	Algorithms
Montemanni et al. [10]	(2005)	exchange and improvement	requests and depot visits order	ACS, 2–OPT
Hanshar[13] Ombuki-Berman	(2007)	route-first cluster-later	requests order	TS, 2–OPT λ-interchange
Hanshar[13] Ombuki-Berman	(2007)	route-first cluster-later	requests order	GA, 2–OPT, greedy insertion
Garrido et al. [15, 19, 20]	(2009)	exchange and improvement	requests order	set of heuristics
Khouadjia et al. [5, 11, 12]	(2010)	cluster-first route-later	requests assignment	PSO, 2–OPT, greedy insertion
Elhasannia et al. [16]	(2013)	exchange and improvement	requests and depot visits order	GA, set of heuristics
Okulewicz Mańdziuk [6, 14]	(2013)	cluster-first route-later	separate requests priorities and multi requests' clusters centers	PSO, 2–OPT, modified Kruskal, greedy insertion
Elhasannia et al. [21]	(2014)	exchange and improvement	requests and depot visits order	GA, greedy insertion
Mańdziuk Żychowski [17]	(2016)	exchange and improvement	requests order	GA, set of heuristics

cross-over operator for the GA [21], enhancing Ant Colony Optimization (ACO) with a Large Neighbourhood Search (LNS) algorithm [16].

The renewed GA and ACO with LNS approaches are worth mentioning, as they are the first methods to present results for the largest benchmark instance, consisting of 385 requests. Unfortunately, those results have been computed on an Intel Core i5 processor, using the same same time limit of 1500 seconds as the Pentium IV in the original work [10], which renders rest of the results incomparable with the original GA and ACS.

A different approach, in terms of the stopping criterion, optimization category and applied metaheuristic, has been taken by Khouadjia et al. [5, 11, 12]. The MEMSO method proposed in those works uses an Adaptive Memory Particle Swarm Optimizer, which utilizes a discretized version of a PSO velocity update procedure, uses a search space of requests-to-vehicles and assignments (falling into the category of *cluster-first route-second* methods) and limits the time of the computation by the number of fitness function evaluations. 2MPSO approach, developed by the authors of this paper [6, 14], falls into the same category, although it uses a standard continuous PSO and a cluster-based heuristic generating initial solutions. Discussion about the possibilities of using differences of the MEMSO and 2MPSO approaches has been presented in [22].

3. Particle Swarm Optimization

PSO is an iterative global optimization metaheuristic method proposed in 1995 by Kennedy and Eberhart [23] and further studied and developed by many other researchers, e.g., [24–26]. The underlying idea of the PSO algorithm consists in maintaining the swarm of particles moving in the search space. For each particle the set of neighboring particles which communicate their positions and function values to this particle is defined. Furthermore, each particle maintains its current position and velocity, as well as remembers its historically best (in terms of solution quality) visited location. More precisely, in each iteration t, each particle i updates its position x_t^i and velocity v_t^i based on the following formulas:

Position update

The position is updated according to the following equation:

$$x_{t+1}^i = x_t^i + v_t^i. \tag{8}$$

Velocity update

In our implementation of the PSO (based on [24, 27]) velocity v_t^i of particle i is updated according to the following rule:

$$\begin{aligned} v_{t+1}^i =& u_{U[0;g]}^{(1)}(x_{best}^{neighbors_i} - x_t^i) + \\ & u_{U[0;l]}^{(2)}(x_{best}^i - x_t^i) + a \cdot v_t^i \end{aligned} \tag{9}$$

where g is a neighborhood attraction factor, $x_{best}^{neighbors_i}$ represents the best position (in terms of optimization) found hitherto by the particles belonging to the neighborhood of the ith particle, l is a local attraction factor, x_{best}^i represents the best position (in terms of optimization) found hitherto by particle i, a is an inertia coefficient, $u_{U[0;g]}^{(1)}$, $u_{U[0;l]}^{(2)}$ are random vectors with uniform distribution from the intervals $[0, g]$ and $[0, l]$, respectively.

2MPSO algorithm presented in this paper uses a Standard Particle Swarm Optimization 2007 (SPSO-07) [27] with a random star neighborhood topology, in which, for each particle, we randomly assign its neighbors, each of them independently, with a given probability[3]. Please note, that we are using SPSO-07 instead of the newer SPSO-11 [27] since we intentionally take advantage of the natural bias towards the search space center which results in greater probability that PSO will choose solutions closer to the center of the search space. Such an effect was observed in the case of the former version of SPSO [28, 29]. Since in our method the search space center is defined at the current-best particle position, the above mentioned bias increases the solutions of finding the particles nearby of this best location.

3.1. Particle Swarm Optimization applications to Vehicle Routing Problems

PSO have been applied to a various models of the VRP: Capacited VRP (CVRP) [30], VRP with Time Windows (VRPTW) [30], Multi-Depots Vehicle Scheduling Problem (MDVSP) [31], Stochastic VRP (SVRP) [32] and Dynamic VRP (DVRP) [5, 6]. In order to apply PSO algorithm to any type of a discrete problem (such as VRP) either the PSO needs to be modified, to operate in a discrete search space, or the problem search space needs to be defined in a way allowing for the application of the continuous PSO's operators.

Although some of the problems, like VRPTW, need additional care for handling time constrains, the design of the search spaces and the PSO operators modifications might be usually transferred between different VRP models.

Discussed PSO approaches have been chosen on the basis of their variety in the design of the search spaces and operator modifications. Khouadjia et al. [12] proposed a constrained discretized version of the PSO's velocity update formula

[3]Please, note that the "neighboring" relation is not symmetrical, i.e. the fact that particle y is a neighbor of particle x, does not imply that x is a neighbor of y.

Table 2: Summary of different methods of applying PSO to VRPs.

Authors	VRP variant	Search space	PSO modifications
Ai, Kachitvichyanokul [30, 33]	VRP(TW)	requests priorities and single requests' clusters centers	none
Marinakis et al. [32]	SVRP	normalized giant tours	normalized velocity
Wang et al. [31]	MDVSP	separate requests priorities and requests-to-vehicles assignment	discretized position
Khouadjia et al. [5, 11, 12]	DVRP	requests-to-vehicles assignment	discretized position
Okulewicz, Mańdziuk [6, 14]	DVRP	separate requests priorities and multi requests' clusters centers	none

and created a VRP solution from a request-to-vehicle assignment vector and a 2–OPT route optimization. Marinakis et al. [32] took an approach similar to the work on GA [13], with the exception of an indirect continuous route encoding instead of a direct discrete one. The solution has been coded as a vector of real numbers, coding the ranks of the requests on a giant TSP tour. While PSO operated in a continuous search space of the requests priorities, its velocity and position update equations have been changed in order to impose the constraints of a search space to a $[0,1]^m$ hypercube. Wang et al. [31] optimized requests-to-vehicles assignment with a discrete PSO and requests order with a continuous PSO, Ai and Kachitvichyanukul [30, 33] utilized a continuous $\mathbb{R}^{m+2n}$ search space encoding both the requests order and assignment to vehicles. The first m coordinates of the vector in that search space define the ranks of the requests, while second $2n$ coordinates define n requests' cluster centers. Each cluster of requests is assigned to a single vehicle. Finally, the authors of this paper [6] used two separate continuous $\mathbb{R}^{2kn}$ and $\mathbb{R}^m$ search spaces, thus dividing the problem into two optimization phases. In the first phase a requests clustering is performed, with k clusters of requests per single vehicle, while in the second phase a requests ordering is performed.

Summary of the search spaces and PSO operators modification is presented in Tab. 2.

4. Two-Phase Multi-Swarm Particle Swarm Optimization for the Dynamic Vehicle Routing Problem

This section presents the execution path of 2MPSO algorithm optimization process during a single time step. Special emphasis is put on presentation of the two independent VRP encodings which induce continuous search spaces for the PSO algorithm.

The 2MPSO algorithm has been developed by the authors over the last few years. It is implemented in C# as a Microsoft .NET application utilizing Windows Communication Foundation services for controlling its independent optimization processes [34]. The VRP encodings have been introduced in [14], together with a single swarm optimization approach to DVRP. This initial PSO-based algorithm has been later enhanced with a multi-swarm optimization, approximation of the number of vehicles on the basis of heuristic solution and a multi-cluster requests-to-vehicles assignment encoding [6]. This paper provides further development of the 2MPSO method, with a direct transfer of candidate solutions from the previous states of the problem (time slices). Additionally, 2MPSO's parameters have been tuned for a better performance on a benchmark set of the DVRP instances.

The optimization system, built according to the 2MPSO design, consists of a set of optimization services. Those independent services are controlled by a single module responsible for providing new states of DVRP and choosing the best found solution at the end of each time step. An optimization process of 2MPSO algorithm utilizes the following modules:

- **Optimizer**: Responsible for handling a single optimization process and ensuring feasibility of a delivered solution,
- **Tree**: Responsible for creating an initial requests-to-vehicles assignments and estimating the number of necessary vehicles,
- **Greedy**: Responsible for repairing infeasible solutions, providing an alternative initial assignments and alternative vehicles' number estimation,
- **2–OPT**: Responsible for routing the requests assigned to the same vehicle in order to make the requests-to-vehicles assignment possible to be evaluated as a VRP solution and to create initial solutions,
- **Solutions transformer**: Responsible for converting discrete solutions into a continuous representation and adapting solutions obtained in the previous time step to the subsequent time slice,
- **PSO**: Responsible for performing a black-box optimization of the requests-to-vehicles assignment (see Section 4.2.1 for requests' cluster centers encoding) and requests order (see Section 4.2.2 for requests' ranks encoding) in continuous search spaces with the use of PSO algorithm.

Figure 2 presents an UML-based activity diagram of the optimization process across the aforementioned modules. 2MPSO is designed as a configurable algorithm, in which only selected optimization techniques might be used. The particular techniques utilized in a given 2MPSO run are selected by Tree, CHist, DHist, 1PSO and 2PSO configuration flags, depicted as guard conditions on execution branches in Fig. 2.

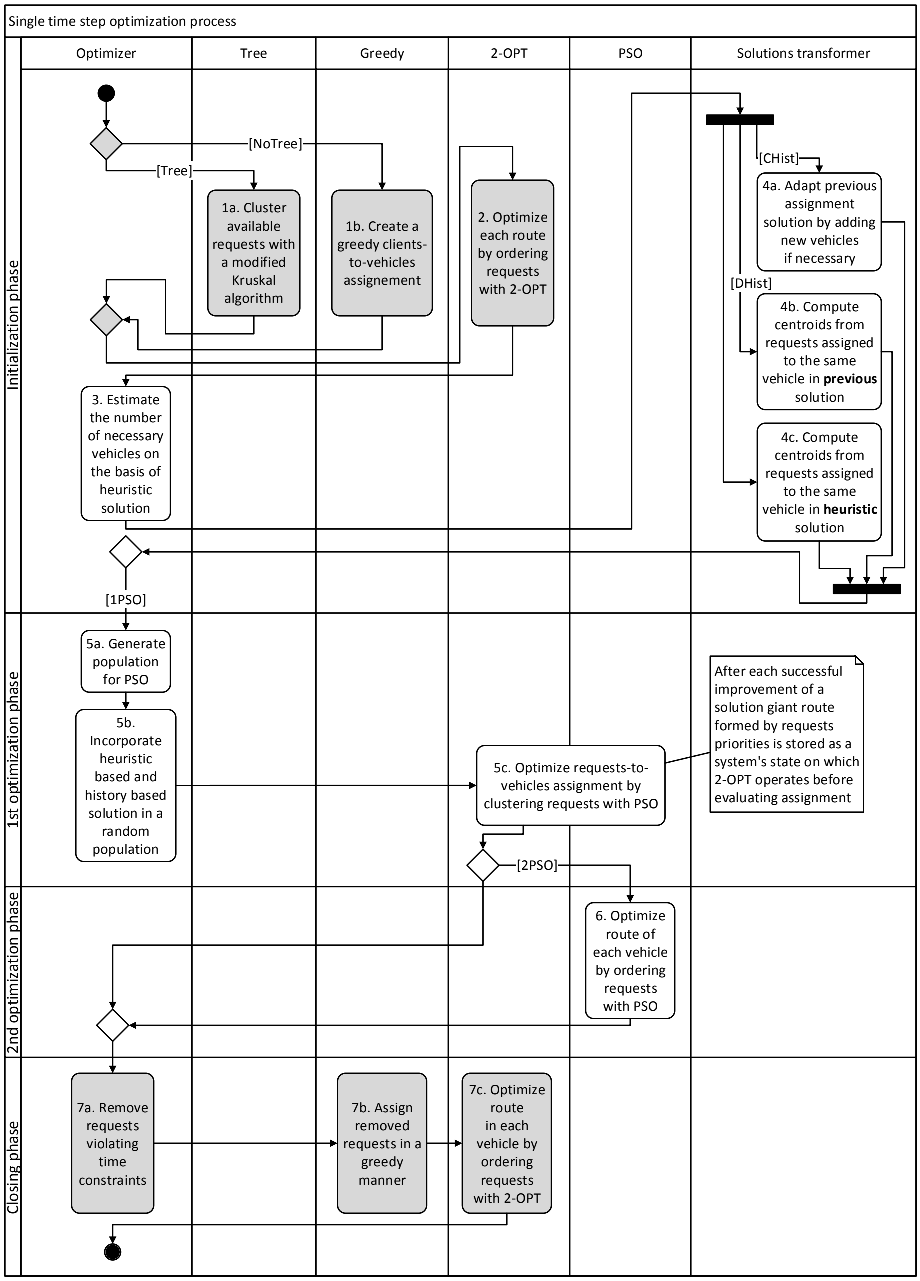


Figure 2: Activity diagram of the optimization process. The obligatory actions are marked with a gray background. Configurable execution branches are labeled with the names used later in the algorithm description and results presentation. For the sake of diagram's readability the "decision node - action node on a single branch - merge node" notation has been simplified to stating the guard conditions, [DHist] and [CHist], directly on the parallel fork branches.

Algorithm 1 Optimization processes' controller pseudo-code for the 2MPSO approach.

V_t a set of vehicles available at time t
C_t a set of requests known at time t which are not ultimately assigned
1: **while** $time \leq end$ **do**
2:   **for all** $optimizer \in optimizers$ **do**
3:     $CreateInitialAndAdaptedSolutions(V_t, C_t)$ {(see Algorithm 3)}
4:     $OptimizeRequestsAssignment(V_t, C_t)$ {(see Algorithm 3)}
5:     $OptimizeVehiclesRoutes(V_t, C_t)$ {(see Algorithm 4)}
6:     $CreateInitialAndAdaptedSolutions(V_t, C_t)$ {(see Algorithm 4)}
7:   **end for**
8:   $bestSolution = ChooseBestSolution(optimizers)$
9: **end while**

4.1. 2MPSO optimization process

The general execution of a whole 2MPSO algorithm proceeds as follows. In each time step, 2MPSO considers the set C_t of requests known at the time t and not ultimately assigned to any vehicle (although they may be tentatively assigned in previous time slice solution). Until the end of the day (line 1 in Algorithm 1) a parallel continuous optimization (line 2 in Algorithm 1) is performed by an ensemble of instances of $optimizers$, which are synchronized at the end of each *time slice*. At the end of each time slice (in line 8 in Algorithm 1) the new $bestSolution$ is chosen among all $optimizerBestSolutions$. In the vehicles' dispatcher module (see Fig. 1) the $bestSolution$ is used to create vehicles' schedules (for the assignments close to the time constraints). The remainder of this section is devoted to a description of a single run of the optimization process (lines 3)–6 in Algorithm 1), with each paragraph guided by a label of an appropriate action node from Fig. 2. Additionally, the description references the appropriate lines in the algorithms' pseudo-code listings.

4.1.1. Initiation phase

(1a) Cluster available requests with a modified Kruskal algorithm [35]. The process starts (line 3 in Algorithm 3) with dividing a set of requests among vehicles as depicted in Algorithm 2. A result of this division is a request-to-vehicles assignment based on the vehicles' capacities.

(1b) Create a greedy clients-to-vehicles assignement. An alternative way to start the process (line 5 in Algorithm 3) is to use a simple greedy algorithm, which processes a random sequence of requests one-by-one, placing them with the lowest insertion cost rule into an existing route or creating a new one if a capacity constraint would be exceed or return-to-depot time constraint violated. A result of the greedy algorithm is a request-to-vehicles assignment and a requests ordering within each of the routes.

Algorithm 2 Pseudo-code for a modified Kruskal algorithm creating a heuristic clients-to-vehicles assignment by solving a capacitated clustering problem.

E set of a weighted edges of a fully connected graph (weight represents distance)
V set of a weighted vertices set on an $\mathbb{R}^2$ plane (weight represents cargo volume)
$CAPACITY$ scalar defining the maximum sum of nodes' weights in a cluster
$DEPOT$ marked node denoting vehicles' depot

$E_{sort} \leftarrow SortByWeightInAscendingOrder(E)$
Initial set of $T_{clusters}$ consists of disjoint single nodes representing unassigned requests and paths representing routes through ultimately assigned requests
$T_{clusters} \leftarrow CreateSeparateTrees(V)$
for all $(v_1, v_2) \in E_{sort}$ **do**
  $Tree(v)$ is a cluster to which node v belongs
  if $Tree(v_1) \neq Tree(v_2)$ **then**
    if $SumNodesW(Tree(v_1)) + SumNodesW(Tree(v_2)) \leq CAPACITY$ **then**
      $T_{clusters} \leftarrow T_{clusters} \setminus \{Tree(v_1), Tree(v_2)\}$
      $T_{clusters} \leftarrow T_{clusters} \cup \{Tree(v_1) \cup Tree(v_2)\}$
    end if
  end if
end for

(2) Optimize each route by ordering requests with 2-OPT. The algorithm described in [36] is applied in order to create a route from an assignment of requests (or improve an existing route from a greedy algorithm). A result of this action is a requests ordering within each of the vehicles' routes.

(3) Estimate the number of necessary vehicles on the basis of heuristic solution. The number of vehicles in the assignment constructed in action (1a) or (1b) is taken as an estimation of a number of vehicles necessary to serve a given set of requests.

The next three actions (4a,4b,4c) are independent of each other and operate on a continuous representation of the requests-to-vehicles assignment presented in detail in Section 4.2.1. An important thing to note is that the size of such a representation is proportional to the number of vehicles used in the DVRP solution.

(4a) Adapt the previous assignment by adding new vehicles if necessary. This action directly passes requests' cluster centers solution from a previous time step and adds random cluster centers if the current estimation of the number of required vehicles is larger than the number used in the input solution.

Algorithm 3 Initial phase and requests-to-vehicles assignment optimization (2MPSO's 1st phase) high-level pseudo-code.

$radius \Leftarrow 2 \max_{l_1, l_2 = k+1, \ldots, k+m} \rho(l_1, l_2)$
heuristicSolution is created in order to increase population diversity and to keep solution in reasonable bounds
if `Tree` **then**
  $heuristicSolution \Leftarrow CapacitatedClustering(C_t)$ {(see Algorithm 2)}
else
  $heuristicSolution \Leftarrow GreedyInsertion(C_t)$
end if
if `1PSO` **then**
  if $t = 0$ **then**
    $bestSolution \Leftarrow Approximate(heuristicSolution)$
  else if `CHist` **then**
    $bestSolution \Leftarrow Adapt(bestSolution)$
  else if `DHist` **then**
    $bestSolution \Leftarrow Approximate(bestSolution)$
  else
    $bestSolution \Leftarrow randomSolution$
  end if
some particles checked by PSO are based on *heuristicSolution* in order to keep *bestSolution* in reasonable bounds
some particles checked by PSO are based on *bestSolution* in order to preserve information about previous solution
all other particles checked by PSO are generated within a *radius* from the $bestSolution$
  $swarm \Leftarrow InitializePSOPopulation(heuristicSolution, bestSolution, radius)$

  for $i = 1, 2, \ldots, maxFirstPhaseIterations\%$ **do**
    $Evaluate(swarm)$
    $UpdateVelocity(swarm)$
    $UpdatePosition(swarm)$
  end for
$optimizerBestSolution$ is treated as a set of vehicles with initial routes
  $optimizerBestSolution = GetBestSolution(swarm)$
else
  $optimizerBestSolution = heuristicSolution$
end if

(4b) Compute centroids from requests assigned to the same vehicle in previous solution. This action computes requests' cluster centers on the sets of the tentatively assigned requests during the previous time step, while ignoring the location of the ultimately assigned ones. It also adds random cluster centers to that solution, as in action (4a).

(4c) Compute centroids from requests assigned to the same vehicle in heuristic solution. This action computes requests' cluster centers on the sets of tentatively assigned requests in a solution created by the heuristic algorithm (action (1a) or (1b)).

4.1.2. 1st optimization phase: requests-to-vehicles assignment

(5a) Generate population for PSO. This action generates a basic population of the PSO, which is a set of random candidate solutions centered within a given $radius$ around the $bestSolution$, which is defined in lines 8–16 in Algorithm 3. The size of a search space is based on the estimated number of necessary vehicles.

(5b) Incorporate heuristic based and history based solution in a random population. This action injects the following solutions into an initial population of the requests-to-vehicle assignment PSO optimizer: a candidate solution based on a $heuristicSolution$ for the current state of the problem, a $bestSolution$ found in the previous time step, a continuous approximation of the $bestSolution$ found in the previous time step (line 17 in Algorithm 3).

(5c) Optimize requests-to-vehicles assignment by clustering requests with PSO. Having the population initialized, the system performs a continuous black-box optimization with the PSO algorithm for $maxFirstPhaseIterations$. While PSO operates in a search space of requests' cluster centers, the generated requests assignments are evaluated as a complete VRP solutions. It is made possible by applying the 2–OPT algorithm in order to construct an optimized route for each vehicle. The output is a VRP solution for a current state of the problem. The solution is encoded as a requests' cluster centers vector (see Section 4.2.1) found by PSO and a requests' ranks vector (see Section 4.2.2) found by 2–OPT.

4.1.3. 2nd optimization phase: requests ordering

(6) Optimize route in each vehicle by ordering requests with PSO. Having the $optimizerBestSolution$ chosen as a result of the previous action (line 23 in Algorithm. 3), its routes are further optimized (lines 2-11). A continuous optimization is performed separately for each vehicle (lines 3-9 in Algorithm 4) for $maxSecondPhaseIterations$.

4.1.4. Closing phase

In some configurations of the 2MPSO algorithm it is possible to obtain a result which is worse than the previous one. Therefore, if the state of the problem did not change, a solution from a previous time step is preserved (line 13 in Algorithm 4).

(7a) Remove requests violating time constraints. The final optimization procedure (line 15 in Algorithm 4), applied to each $optimizerBestSolution$, is aimed at repairing unfeasible routes (violating time constraint from eq. (4)) by means of greedy reassignment of a rearmost requests from such a route. The results of this action are an incomplete VRP solution and a set of unassigned requests.

Algorithm 4 Vehicles' routes optimization (2MPSO's 2nd phase) and closing phase high-level pseudo-code.

1: **for all** $vehicle \in optimizerBestSolution$ **do**
2:   **if** `2PSO` **then**
3:     $swarm \Leftarrow InitializePSOPopulation(vehicle.route)$
4:     **for** $i = 1, 2, \ldots, maxSecondPhaseIterations$ **do**
5:       $Evaluate(swarm)$
6:       $UpdateVelocity(swarm)$
7:       $UpdatePosition(swarm)$
8:     **end for**
9:     $vehicle.route \Leftarrow GetBestRoute(swarm)$
10:   **end if**
11: **end for**
12: **if** $C_t \subseteq C_{t-1}$ AND $optimizerBestSolution > bestSolution$ **then**
if the requests set has not changed between t and $t-1$, we preserve the $bestSolution$ if it was better
13:   $optimizerBestSolution \Leftarrow bestSolution$
14: **else**
reassign in a greedy way all requests violating time constraint of the problem (see eq. (4))
15:   $optimizerBestSolution = RepairBestSolution(optimizerBestSolution)$
16: **end if**
17: **for all** $vehicle \in optimizerBestSolution$ **do**
18:   $vehicle.route \Leftarrow EnhanceWith2OPT(vehicle.route)$
19: **end for**

(7b) Assign removed requests in a greedy manner. In order to get a complete VRP solution, the unassigned requests are inserted into an incomplete VRP solution obtained as a result of the previous action in the same manner as described in action (1b).

(7c) Optimize a route of each vehicle by ordering requests with 2-OPT. To get a finally polished result, the routes are additionally optimized with a 2–OPT algorithm (line 18 in Algorithm 4).

4.2. VRP encoding and fitness functions

As already stated, 2MPSO uses PSO algorithm in both phases of the optimization of the current state of the DVRP problem. Therefore, two independent types of continuous encodings are introduced in this subsection, together with appropriate fitness functions and swarms initialization methods.

4.2.1. Requests-to-vehicles assignment encoding

Particles positions' in the first phase denote centers of clusters of requests assigned to certain vehicles. The area of clients' requests locations is divided among vehicles on the basis of the Euclidean distances from the client's location to the cluster centers (i.e. a request is assigned to a vehicle which serves the nearest cluster). The number of clusters k assigned to each vehicle is a parameter of the 2MPSO algorithm.

The solution vector $(v_1.x_1, v_1.y_1, v_2.x_1, \ldots, v_m.y_1, v_1.x_2, v_1.y_2 \ldots, v_m.y_k)$ for the m vehicles and k requests' clusters per vehicle is transformed into a VRP solution in the following way:

1. Distances between $(v_i.x_j, v_i.y_j)$ and all not decisively assigned clients' locations are computed.
2. The computed distances (treated as client-vehicle edges) are sorted in the ascending order.
3. The algorithm iterates over the sorted distances assigning each client to a vehicle with a nearest cluster center until all the clients have been assigned.
4. The requests assigned to a single vehicle are formed in a random route and reordered with the use of 2–OPT algorithm [36].

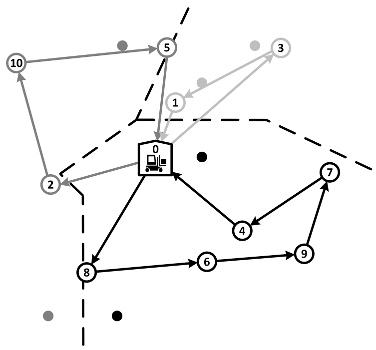
The example depicted in Fig. 3 consists of 3 vehicles with 2 requests cluster centers per vehicle. The division of an $\mathbb{R}^2$ plane, on which the requests are located, imposes their following assignment: v_1 operates in the upper-right part of the plane and has been assigned set containing 2 requests ($\{1, 3\}$), v_2 operates in the middle-right and lower-right parts of the plane and has been assigned set containing 5 requests ($\{4, 6, 7, 8, 9\}$), v_3 operates in the lower-left and upper-left parts of the plane and has been assigned set containing 3 requests ($\{2, 5, 10\}$). Subsequently 2–OPT algorithm created routes from those sets of requests.

After performing such clustering and routing procedure the $COST(r_1, r_2, \ldots, r_n)$ function (see eq. (1)) may be directly applied as a fitness function for particles evaluation. In order to promote finding feasible solutions by the PSO, a penalty function is added to the DVRP cost function. The penalty is a sum of squares of times of late returns to the depot (with I being an indicator function):

$$PENALTY(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} I(arv_{r_{i,m_i}} > t_{end})(arv_{r_{i,m_i}} - t_{end})^2 \qquad (10)$$

4.2.2. Vehicle's route encoding

In the second phase of the 2MPSO, the clients-to-vehicles assignment remains unchanged while the order of visits is optimized. In this phase each individual encodes the order of requests assigned to a given vehicle (each vehicle's route is optimized by a separate PSO instance). The sequence of visits is obtained by sorting indices of each of the proposed solution vector in the ascending order by their values. The example depicted in Fig. 3 consists of 3 routes: 0–3–1–0, 0–8–6–9–7–4–0 and 0–2–10–5–0. Those routes have been constructed by fixing the requests-to-vehicles assignment obtained in the first phase of optimization and sorting the indexes of requests vector within each vehicle by its values: $(3 : 0.5, 1 : 1.0), (8 : 0.2, 6 : 0.4, 9 : 0.6, 7 : 0.8, 4 : 1.0), (2 : 0.3, 10 : 0.7, 5 : 1.0)$.



requests cluster centers (1st phase encoding)

1st vehicle		2nd vehicle		3rd vehicle		1st vehicle		2nd vehicle		3rd vehicle	
1.0	1.4	1.0	0.0	-0.5	2.1	2.0	2.1	-0.6	-3.0	-1.9	-3.0

requests ranks (2nd phase encoding)

1.0	0.3	0.5	1.0	1.0	0.4	0.8	0.2	0.6	0.7

Figure 3: Example of a VRP with 3 vehicles and 10 clients' requests. Solid lines represent possible routes, whose lengths are used as an evaluation function by 2MPSO. Dotted lines separate the operating areas assigned to vehicles. Operating areas are defined by their cluster centers denoted by $(v_i.x_j, v_i.y_j)$ for the ith vehicle and the jth cluster center. The division of the $\mathbb{R}^2$ plane imposes the requests-to-vehicle assignment. The particular routes are defined by the requests ranks vector, by sorting requests identifier by their rank within each vehicle.

The solution assessment in the second phase is equal to the length of a route for a given vehicle, defined by the proposed ordering. The final cost value is equal to the sum of the assessments of the best solutions found by each of the second-phase optimization algorithm instances. The same type of square penalty function (see eq. (10)) as in the first phase optimization is applied to each of the routes if necessary.

4.2.3. Solution transfer for swarm initialization

In order to take advantage of existence of the candidate solutions for the previous state of the DVRP and a heuristic solution for the current state of the problem obtained by capacitated clustering (or greedy insertion) and 2–OPT algorithm, 2MPSO uses those solutions to initialize particles locations. The previous requests clusters centers solution may adapted in two ways: a direct transfer and a continuous approximation.

Directly transferred solution consists of a vector of requests' clusters centers obtained in a previous time slice, with additional random requests' cluster centers added if the current state of the problem is estimated by a heuristic solution as needing additional vehicles. In the continuous approximation approach all the clusters centers for a given vehicle are set to the average location (with a small random perturbation) of the tentatively assigned requests within that vehicle. The same average location method is applied to the heuristic solution.

The order of the requests is passed directly in both of the transfer methods, with new requests being initialized with a random rank. For the solutions obtained by the 2–OPT, in the first phase of optmization, a continuous representation for the second phase is created by setting the ranks within the given vehicle from the best found requests assignment to a following sequence: $(\frac{1}{m_i}, \frac{2}{m_i}, \ldots, \frac{m_i}{m_i})$.

Table 3: Parameter values in the baseline experiments.

Parameter	Value(s)	
	FFE budget	Time budget
DVRP simulation		
T_{co}	0.5	0.5
n_{ts}	40	40
T_{ac}	4%	4%
2MPSO		
#clusters per vehicle	2	2
#parallel optimization processes	8	8
#particles	22	22
#iterations 1st/2nd phase	140/0	1.875/0 sec.
PSO		
g	0.60	0.60
l	2.20	2.20
a	0.63	0.63
$P(X \text{ is a neighbor of } Y)$	0.50	0.50

In both phases the swarm is initialized within a certain radius around the previous best solution with some of the particles placed at the exact locations of transferred solutions.

5. Experiments and 2MPSO results

5.1. Benchmark files

In order to evaluate the performance of the algorithm we used dynamic versions of Christofides' [37], Fisher's [38] and Taillard's [39] benchmark sets adapted to the DVRP by Kilby et al. [8]. Each instance consists of between 50 and 385 requests to be served by a fleet of 50 vehicles (the number of requests is a part of the benchmark's name). The chosen benchmarks are very popular in DVRP literature and, in particular, were used in all papers we make a comparison with in this study. Generally speaking, the benchmark sets are very diverse. They include examples of a very well clustered problems, semi-clustered ones, and completely unstructured instances. Also the volume distribution (especially its skewness) significantly differs across the benchmarks.

Visualization of requests' distribution for all the benchmarks can be found at our project website [40] with some examples presented in Appendix B.

5.2. 2MPSO parameters

The main parameters for the baseline experiments are presented in the Table 3. For PSO g, l, a and P where chosen experimentally based on some number of initial tests. The stopping criterion was defined based on either the number of fitness function evaluations (FFE) or computational time limit. The

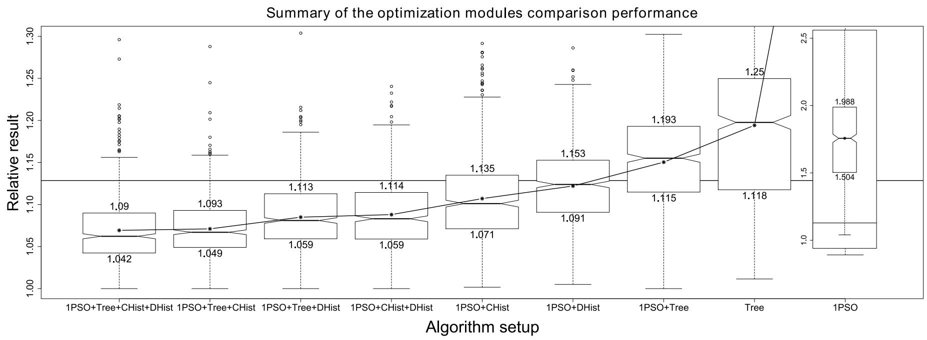


Figure 4: Distribution of the relative results for 2MPSO configurations with various optimization components switched off run with the FFEs number computations limit. Horizontal line depicts the average result of a MEMSO algorithm.

limit for the total number of FFE and the cut-off time were imposed according to the former literature results. The total time limit of 75 seconds, on multithreaded Intel Core i7, for 2MPSO has been proposed and used as an equivalent to the time limit of 750 seconds for GA, which has been run on a single threaded Intel Pentium IV machine [13]. The number of parallel optimization processes stemmed from the number of virtual CPUs on the testing machine. The number of cluster per vehicle, number of time slices, the advanced commitment time, and the ratio of the number of solutions to the number of iterations were experimentally tuned on the set of 21 benchmarks used in the FFE limited experiments[4].

5.3. Experiments setup

In order to achieve statistically significant and comparable results each benchmark has been solved 30 times with the common parameter set. From those 30 runs the average value (showing the general quality of the algorithm) and the minimum value (showing the potential for achieving high quality solutions) were computed. The significance of the average results has been tested against the literature results with the Student's *t*-test.

In order to summarize all the results for a given parameter set the particular results were normalized by dividing them by the shortest known solution for a given benchmark instance. Such normalized results are referred to (on the boxplots) as *relative results*.

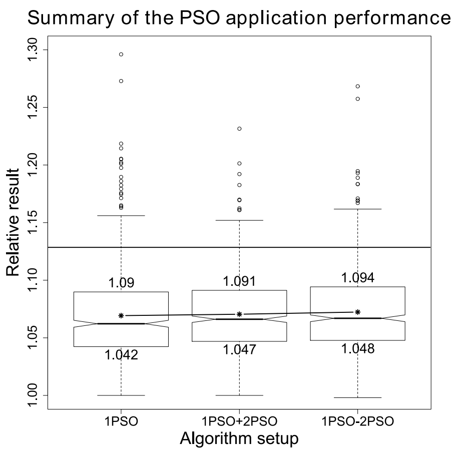
5.4. Optimization modules analysis

The critical research question of this paper is to estimate the particular 2MPSO's components contribution on achieving good results on the DVRP

[4]The details and results of the parameter tuning procedure are presented in Appendix A.

Table 4: Summary of the average results on a set of benchmark instances for dividing optimization budget between phases. Percentages denote the division of the number of fitness function evaluations between 1PSO and 2PSO. Best result is marked in bold and grey background denotes results which are not significantly worse than the best average (based on the Student's t-test).

	$1PSO(100\%)$	$1PSO(86\%)$ $+2PSO(14\%)$	$1PSO(86\%)$
	Avg	Avg	Avg
c50	**578.31**	579.65	581.62
c75	**903.72**	905.72	907.64
c100	933.46	**926.56**	931.43
c100b	845.8	843.52	**841.15**
c120	**1071.38**	1081.9	1090.51
c150	**1134.2**	1146.07	1143.32
c199	1408.7	1415.27	**1407.89**
f71	298.5	**291.04**	291.24
f134	11892	11904.48	**11880.04**
tai75a	**1805.03**	1818.46	1844.97
tai75b	1422.6	1419.48	**1408.43**
tai75c	1510	**1504.51**	1513.15
tai75d	**1433.25**	1438.07	1441.79
tai100a	**2216.23**	2235.43	2259.82
tai100b	**2136.8**	2162.21	2142.82
tai100c	**1494.72**	1497.99	1499.23
tai100d	**1727.95**	1736.32	1746.72
tai150a	**3530.82**	3558.19	3558.34
tai150b	3026.89	3046.21	**3019.01**
tai150c	2603.53	**2583.87**	2611.98
tai150d	3009.01	**2985.12**	2995.35
sum	**44982.9**	45080.07	45116.45



benchmark set. For testing various configurations of the 2MPSO, two types of experiments have been conducted. The first one was performed to assess the need of the second phase PSO optimization and its results are presented in Table 4. The second one has been performed to assess the impact on DVRP results of introducing heuristic solutions, directly transferred solutions, approximately transferred solutions and the PSO optimization itself. Results of that experiment are presented in Figure 4 and Table 5. Discussion of the particular techniques uses the configuration flags (Tree, CHist, DHist, 1PSO, 2PSO) introduced in Section 4.1, while describing the activities of the 2MPSO algorithm.

5.4.1. 2PSO: 2nd phase PSO optimization

The analysis of the results presented in Table 4 points out the relative insignificance of the continuous optimization performed in the second phase (likely due to using 2–OPT in the first phase). While having a second phase PSO optimization slightly improves the average results (cf. columns $1PSO(86\%) + 2PSO(14\%)$ and $1PSO(86\%)$), a visibly better improvement, although not statistically significant, can be achieved if the same total budget of FFEs is spent exclusively in the 1st phase (cf. columns $1PSO(100\%)$ and $1PSO(86\%) + 2PSO(14\%)$).

Therefore, in all further experiments the whole metaheuristic optimization computations budget is utilized by the 1st phase PSO with a baseline configuration (denoted as 1PSO or simply 2MPSO) having all the other configuration flags (Tree, CHist, DHist) switched on.

Table 5: Summary of the average results on a set of benchmark instances for various optimization modules switched off. Best result is marked in bold and grey background denotes results which are not significantly worse than the best average (based on the Student's t-test).

	1*PSO* +*Tree* +*CHist* +*DHist*	1*PSO* +*Tree* +*CHist*	1*PSO* +*Tree* +*DHist*	1*PSO* +*CHist* +*DHist*	1*PSO* +*CHist*	1*PSO* +*DHist*	1*PSO* +*Tree*	*Tree*	1*PSO*
	Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg
c50	578.31	580.6	**576.48**	584.25	584.78	587.08	605.64	736.66	768.49
c75	**903.72**	905.32	906.5	927.1	947.61	925.99	999.11	1121.94	1422.61
c100	933.46	**925.54**	930.17	939.31	975.83	970.03	1014.72	1102.27	1575.53
c100b	845.8	842.88	838.74	843.89	852.98	853.19	838.36	**835.05**	982.56
c120	**1071.38**	1090.56	1085.82	1119.76	1097.47	1124.65	1087.68	1100.35	1417.92
c150	**1134.2**	1154.63	1145.67	1193.8	1278.92	1233.69	1243.95	1278.69	2187.39
c199	1408.7	1421.52	**1403.86**	1441.17	1488.73	1537.94	1608.39	1640.37	2528.06
f71	298.5	**290.26**	294.57	291.23	298.68	297.8	301.94	332.05	402.32
f134	11892	**11878.62**	11992.36	11974.4	11982.38	12192.3	12188.79	12928.45	12399.6
tai75a	1805.03	**1804.5**	1828.85	1823.18	1912.45	1868.8	1916.67	2011.77	2862.4
tai75b	1422.6	**1418.35**	1485.56	1423.05	1431.05	1523.09	1535.72	1605.2	2050.18
tai75c	1510	**1501.34**	1520.98	1541.31	1552.97	1557.35	1594.42	1665.86	2094.51
tai75d	**1433.25**	1452.19	1454.6	1442.08	1452.63	1487.34	1496.25	1480.29	2444.79
tai100a	**2216.23**	2250.34	2277.29	2308.59	2330.15	2382.92	2515.87	2582.84	4365.11
tai100b	**2136.8**	2153.72	2165.55	2194.7	2259.7	2289.77	2356.51	2356.64	4187.3
tai100c	**1494.72**	1501.72	1536.52	1517.49	1524.9	1558.33	1630.28	1570.9	2382.15
tai100d	**1727.95**	1737.12	1757.42	1769.5	1779.56	1810.48	1953.39	2063.46	3004.16
tai150a	3530.82	**3522.7**	3718.11	3540.63	3707.63	3845.85	3956.46	3735.37	8078.31
tai150b	**3026.89**	3048.36	3096.83	3135.83	3170.44	3193.1	3274.73	3472.06	6373.7
tai150c	2603.53	**2580.25**	2698.73	2675.97	2669.49	2837.01	2833.65	2678.9	4301.98
tai150d	3009.01	**2995.86**	3090.26	3061.67	3081.2	3172.15	3251.66	3354.67	5586.19
sum	**44982.9**	45056.38	45804.87	45748.91	46379.55	47248.86	48204.19	49653.79	71415.26

5.4.2. 1PSO: 1st phase PSO optimization

Analysis of the results presented in Figure 4 and Table 5 shows that the continuous optimization of the requests-to-vehicles assignment is crucial for obtaining high quality solutions (cf. results of configurations using 1PSO with any of the additional modules vs. a Tree only configuration). It is also important to observe that PSO initialized only with random solutions (1PSO configuration) in each time step provides very low quality solutions and needs at least one type of "reasonably good" solution to perform better than a discrete capacitated clustering on its own.

5.4.3. CHist and DHist: solutions transfer between problem states

Utilizing at least one type of a solution transfer between subsequent problem states is important for obtaining high quality solutions (1PSO+Tree, Tree and 1PSO configuration, which do not have any type of solutions transfer obtained the worst results). From the two types of a transfer a direct passing of the solution (CHist flag) is more important than passing its approximation (DHist flag). For some benchmark instances 1PSO+Tree+CHist configuration obtained even better average solutions than the baseline 1PSO+Tree+CHist+Dhist setup. While considering the overall average performances, the setup without a given type of transfer always gives worse results than the one utilizing it.

Table 6: Comparison of the GA, TS, EH algorithms with a 2MPSO algorithm. Time limit of 75 seconds has been chosen in order to make results comparable with those obtained on an Intel Pentium IV. The best minimum and average values within each setup are bolded and statistically significant differences between authors' and literature results are marked with a grey background. The statistical significance has been measured by a one-sided t-tests with $\alpha = 0.05$.

	GA [13] 750 seconds Intel Pentium IV @2.8GHz		*TS* [13] 750 seconds Intel Pentium IV @2.8GHz		*EH* [15] 250 seconds Athlon 64 @2.2 GHz		*ACOLNS* [16] 1500 seconds Intel Core i5 @2.4 GHz		2*MPSO* 75 seconds Intel Core i7(2nd) @3.4GHz	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	570.89	593.42	603.57	627.90	597.72	632.71	601.78	623.09	**562.70**	**581.46**
c75	981.57	1013.45	981.51	1013.82	979.29	1019.05	1003.20	1013.47	**874.08**	**905.95**
c100b	881.92	900.94	891.42	932.14	956.67	1020.02	932.35	943.05	**819.56**	**844.90**
c100	961.10	987.59	997.15	1047.60	975.20	1003.95	987.65	1012.30	**882.96**	**930.95**
c120	1303.59	1390.58	1331.80	1468.12	1245.94	1372.45	1272.65	1451.60	**1066.15**	**1085.46**
c150	1348.88	1386.93	1318.22	1401.06	1342.91	1413.05	1370.33	1394.77	**1147.50**	**1195.95**
c199	1654.51	1758.51	1750.09	1783.43	1689.55	1747.02	1717.31	1757.02	**1434.70**	**1503.94**
f71	301.79	309.94	280.23	306.33	287.99	299.58	311.33	320.00	**270.35**	**290.62**
f134	15528.81	15986.84	15717.90	16582.04	14801.60	14952.66	15557.82	16030.53	**11773.74**	**12038.02**
tai75a	1782.91	1856.66	1778.52	1883.47	1769.75	1859.25	1832.84	1880.87	**1767.64**	**1825.87**
tai75b	1464.56	1527.77	1461.37	1587.72	1450.45	1502.09	1456.97	1477.15	**1366.80**	**1419.66**
tai75c	1440.54	1501.91	**1406.27**	1527.80	1685.10	1779.08	1612.10	1692.00	1427.76	**1487.39**
tai75d	**1399.83**	**1422.27**	1430.83	1453.50	1432.92	1445.89	1470.52	1491.84	1404.75	1442.45
tai100a	2232.71	2295.61	2208.85	2310.37	2227.43	2309.90	2257.05	2331.28	**2196.91**	**2261.66**
tai100b	2147.70	2215.39	2219.28	2330.52	2183.38	2221.40	2203.63	2317.30	**2060.46**	**2151.73**
tai100c	1541.28	1622.66	1515.10	1604.18	1656.97	1756.25	1660.48	1717.61	**1476.24**	**1512.13**
tai100d	1834.60	1912.43	1881.91	2026.76	1834.47	2029.45	1952.15	2087.96	**1676.10**	**1746.44**
tai150a	**3328.85**	3501.83	3488.02	3598.69	3346.02	**3487.78**	3436.40	3595.40	3476.48	3777.98
tai150b	2933.40	3115.39	3109.23	3215.32	**2874.72**	**3068.64**	3060.02	3095.61	2978.30	3120.09
tai150c	2612.68	2743.55	2666.28	2913.67	2583.13	2731.14	2735.39	2840.69	**2532.23**	**2678.16**
tai150d	**2950.61**	**3045.16**	2950.83	3111.43	3084.58	3252.03	3138.70	3233.39	2958.75	3141.63
tai385	NA	NA	NA	NA	NA	NA	33062.06	35188.99	**31162.15**	**32801.70**

5.4.4. Tree: capacitated clustering

Capacitated clustering by a modified Kruskal algorithm proved to be a reasonably well performing algorithm in itself (especially if one considers the fact that it processes the whole sequence of problem states within a few seconds, even for the larger instances). Analysis of the average results raises a similar conclusion to that of the solution transfer: configuration without the capacitated clustering perform worse than the corresponding ones utilizing it.

5.5. Comparison with the literature results

The analysis of the configurations of the 2MPSO resulted in selecting a 1PSO+Tree+CHist+DHist configuration as a baseline one, which is simply referred to as 2MPSO, in the comparison with the literature results.

As already mentioned, two types of comparison with the literature results were made: the number of FFE limited experiment and the computations time limited experiment.

The comparison of the time limited approaches includes the algorithms providing state-of-the art results for at least one benchmark problem. The results of the experiment with 2MPSO's computations time limited to 1.875 for each of the 40 time slices are presented in Tab. 6. It can be observed that 2MPSO obtained 18 out of 22 best average results (all of them statistically significantly better), including the largest `tai385` benchmark.

Table 7: Comparison of the MAPSO and MEMSO algorithms with the initial and current version of the 2MPSO algorithm. Total number of fitness function evaluation of 10^6 has been chosen in order to make the results comparable. The best minimum and average values within each setup are bolded and statistically significant differences between authors' and literature results are marked with a grey background. The statistical significance has been measured by a one-sided t-tests with $\alpha = 0.05$.

	MAPSO [12] $(25*8*(0.5*10^4))$		*MEMSO* [5] $(25*8*(0.5*10^4))$		2*MPSO* 2014 [6] $(25*8*(1.4*10^4))$		2*MPSO* $(40*8*(0.31*10^4))$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	571.34	610.67	577.60	592.95	583.09	618.59	**544.11**	**578.31**
c75	931.59	965.53	928.53	962.54	904.83	946.85	**884.43**	**903.72**
c100b	866.42	882.39	864.19	878.81	830.58	875.47	**819.56**	**845.80**
c100	953.79	973.01	949.83	968.92	926.10	966.27	**902.00**	**933.46**
c120	1223.49	1295.79	1164.63	1284.62	1061.84	1176.38	**1053.18**	**1071.38**
c150	1300.43	1357.71	1274.33	1327.24	1132.12	1208.60	**1098.03**	**1134.20**
c199	1595.97	1646.37	1600.57	1649.17	1371.61	1458.01	**1362.65**	**1408.70**
f71	287.51	296.76	283.43	**294.85**	302.50	319.01	**274.16**	298.50
f134	15150.50	16193.00	14814.10	16083.82	11944.86	12416.65	**11746.40**	**11892.00**
tai75a	1794.38	1849.37	1785.11	1837.00	1721.81	1846.03	**1685.23**	**1805.03**
tai75b	1396.42	1426.67	1398.68	1425.80	1418.82	1451.92	**1365.36**	**1422.60**
tai75c	1483.10	1518.65	1490.32	1532.45	1456.90	1560.68	**1439.02**	**1510.00**
tai75d	1391.99	**1413.83**	**1342.26**	1448.19	1445.58	1481.25	1408.79	1433.25
tai100a	2178.86	2214.61	2170.54	**2213.75**	2211.30	2327.20	**2137.30**	2216.23
tai100b	2140.57	2218.58	2093.54	2190.01	**2052.54**	**2131.91**	2060.65	2136.80
tai100c	1490.40	1550.63	1491.13	1553.55	1465.06	1519.44	**1458.81**	**1494.72**
tai100d	1838.75	1928.69	1732.38	1895.42	1722.16	1808.67	**1663.87**	**1727.95**
tai150a	3273.24	3389.97	**3253.77**	**3369.48**	3367.55	3537.81	3338.71	3530.82
tai150b	**2861.91**	**2956.84**	2865.17	2959.15	2911.22	3033.83	2910.06	3026.89
tai150c	2512.01	2671.35	2510.13	2644.69	2510.51	**2579.72**	**2497.65**	2603.53
tai150d	**2861.46**	**2989.24**	2872.80	3006.88	2893.54	2992.53	2869.79	3009.01

The results of the experiment with computations budget bound by the number of FFEs are compared with the state-of-the art MAPSO and MEMSO approaches utilizing a discrete version of the PSO algorithm. The results of the computations with the number of FFEs limited to 3125 within each time slice for each of the 8 parallel optimization processes is presented in Tab. 7. It can be observed that 2MPSO obtained 15 out of 21 best average results (13 of them statistically significantly better), with MAPSO and MEMSO approaches remaining competitive for the Taillard's benchmark set.

6. Conclusions

The 2MPSO algorithm presented in this article outperforms other literature approaches, both in the time bounded computations and in those bounded by a number of FFEs. In the time bounded experiment 2MPSO outperforms the average length of the **GA**'s routes by **7.1%** and **ACOLNS**'s by **10.4%**. In the FFE bounded experiment 2MPSO outperforms the average length of **MEMSO**'s routes by **5.7%**. The average results of the **2MPSO** implementation presented in this paper outperform on its initial version (from the year 2014) [6] by **3.2%**.

Detailed analysis of the optimization techniques used by our algorithm confirms the findings from [4], on importance of including both the previous and random solutions in the initial population after the problem state change. In the area of the initial population composition, it also proved beneficial to add solution based on a heuristic clustering algorithm, with the most credit for the quality of the final solution belonging to the assignment optimization performed by the PSO.

In the area of the knowledge transfer between problem states, our research experimentally proves that direct transfer of solutions, without discretizing them, improves the results obtained on a benchmark set.

Finally, optimizing vehicles routes with the PSO, after they have been already optimized with a 2–OPT algorithm in assignment phase proved to be unnecessary. The computations budget is more efficiently utilized by optimizing requests-to-vehicles assignment.

Therefore, the success of the 2MPSO method can be contributed to three crucial factors: (1) the usage of the modified Kruskal algorithm, (2) knowledge transfer between consecutive time slices by means of transferring the best solution from the previous time slice, and (3) the use of continuous optimization meta-heuristic, in particular in requests-to-vehicles-assignment phase.

In the future work we plan to test other continuous optimization algorithms (eg. Differential Evolution) as a main optimization engine, instead of the PSO. Additionally, we are going to analyse the reason for obtaining good quality final results by solving DVRP a series of dependent static VRP instances, despite insufficient initial knowledge on the properties of the final set of requests. Also, we plan to apply a robust-like optimization approach, to directly account for a dynamic characteristics of the DVRP.

Acknowledgement

The research was financed by the National Science Centre in Poland, grant number DEC-2012/07/B/ST6/01527.

Appendix A. Parameter tuning

This appendix presents the results of tuning two major parameters of the proposed optimization method: the *number of time slices* and the *advanced commitment time*. In addition, the ratio of the iterations to the number of individuals in the population has been tuned. Finally, the impact of a number of clusters and knowledge transfer type on a quality and computations time has been measured for experiments with the fixed FFE budget.

Appendix A.1. Advance commitment time and number of time slices

In order to choose the appropriate *number of time slices* and the size of time buffer (*advanced commitment time*) the following tests were run for all benchmark instances:

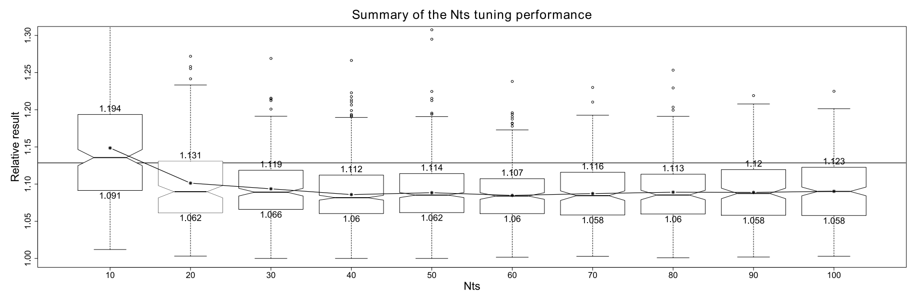


Figure A.5: Relative performance of the 2MPSO algorithm on all benchmark files for various numbers of *time slices* (with the same FFE budget and the same population size to iterations ratio). Horizontal line depicts the average result of a MEMSO algorithm.

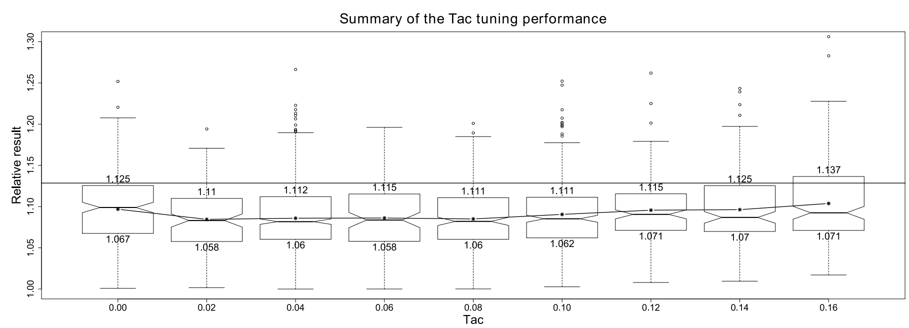


Figure A.6: Relative performance of the 2MPSO algorithm on all benchmark files for various *advanced commitment time* values. Horizontal line depicts the average result of a MEMSO algorithm.

- 2MPSO with the population size to iterations ratio equal to $1 : 6.25$ ($20 : 250$), the *advanced commitment time* equal to 0.04 and the *number of time slices* from the set $\{10, 20, \ldots, 100\}$;
- 2MPSO with the population size to iterations ratio equal to $1 : 6.25$ ($20 : 250$), the *number of time slices* equal to 40, and the *advanced commitment time* from the set $\{0, 0.02, \ldots, 0.16\}$.

The results of testing various *numbers of time slices* and *advanced commitment times* are presented in Figures A.5 and A.6, respectively. Since there was no significant difference in the distribution of relative results between the experiments with 40 and more time slices, the *number of time slices* was set to 40 in the main experiments presented in the paper. The *advanced commitment time* was set to 0.04 as no relevant difference was observed between the *advanced commitment time* set to 0.04 and 0.08, and smaller time buffer is a more intuitive choice.

Appendix A.2. Population size to iterations ratio

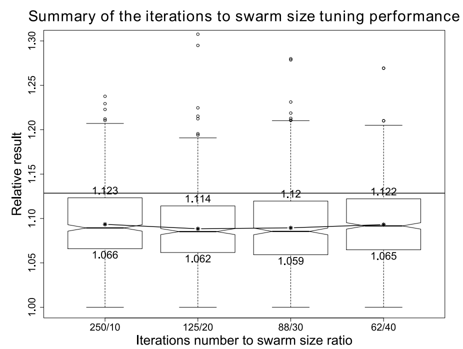


Figure A.7: Relative performance of the 2MPSO algorithm on all benchmark files for various population size to iterations ratios with the constant FFE budget. Horizontal line depicts the average result of a MEMSO algorithm.

In order to choose the efficient number of iterations to population size ratio an experiment with the *advanced commitment time* equal to 0.04, the *number of time slices* equal to 50[5] and the population size from the set $\{10, 20, 30, 40\}$ has been conducted. Figure A.7 presents the relative results for various population size to iterations ratios for constant budget of 10^6 FFE per *optimizer* (population/swarm). There seems to be no significant difference between the distribution of relative results for swarm sizes equal to 20 and 30. Therefore, in the main experiment, the number of iterations to population size ratio was set to 6.25 : 1 (125 : 20).

Appendix A.3. Knowledge transfer and number of requests clusters per vehicle

Final parameter tuning experiment concerned the optimum number of requests clusters for both type of knowledge transfer. Experiments with transferring previous solution through a discretization phase are denoted by *Hist*, while experiments with direct transfer of continuous solution are denoted by *CHist*. The experiments have been conducted for the number of clusters $k = 1, 2, 3$ and resulted in choosing $k = 2$ as a value for the baseline experiments. One cluster encoding seems to be beneficial if the limit on computations time becomes an optimization process constraint. Three clusters proved to generate to large search space to be effectively utilized.

Appendix B. Selected benchmark instances and obtained best results

In section Appendix B.1 of this appendix the differences in spatial and volume distributions of requests between example benchmark instances are pre-

[5]This was the first parameter tuning experiment, therefore the initial number of 50 *time slices* has been used.

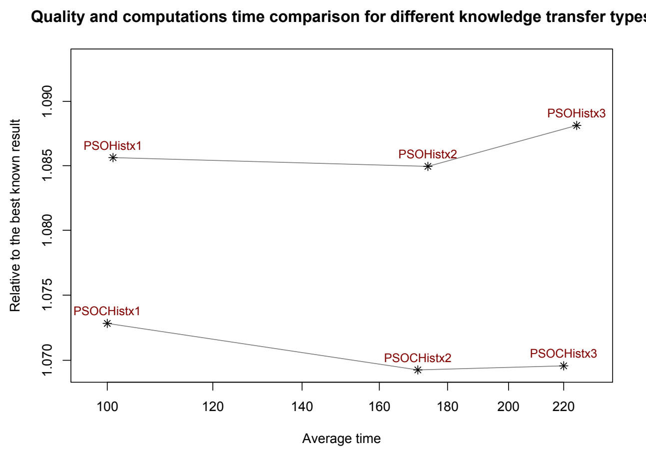


Figure A.8: Average relative performance and average computations time for direct (CHist) and indirect (Hist) previous solutions transfer, with xk denoting the number of requests' clusters used in the encoding of the requests-to-vehicles assignment.

sented, with the aim of pointing the probable explanation of better performance of the discrete encoding based algorithms on some of the instances.

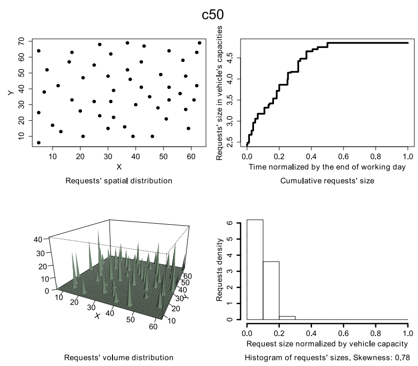
In section Appendix B.2 depiction and schedules of solutions obtained by the 2MPSO algorithm for the chosen benchmark instances are presented. Up to date best obtained solutions can be found at our website [40].

Appendix B.1. Benchmark instances

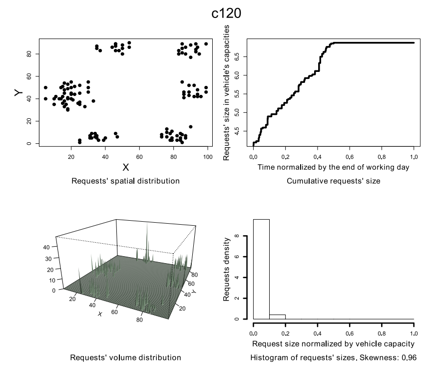
Spatial and volume distributions, as well as histogram of requests' sizes and the plot of their cumulative size over time, of the four exemplar benchmark sets are presented in Figs. B.9a–B.9d. The distributions of requests' sizes differ mainly in their skewness and the existence of relatively large requests. The spatial distribution of requests varies from uniform-like (e.g. c50) to clearly structured ones (e.g. c120, tai150b).

Appendix B.2. Obtained results

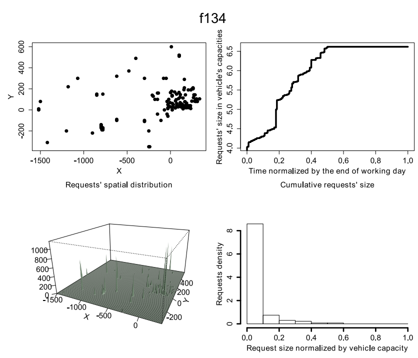
Selected results obtained by the 2MPSO algorithm are presented in Figures B.10a–B.10d. Tables B.8–B.11 present the schedules for the fleet of vehicles in terms of the vehicle id, request id, its location (columns X, Y), time of availability in the system (column $Known$) and scheduled visit time (column $Time$). The requests are grouped by vehicle id and ordered by scheduled time. Horizontal lines denote returns to a depot.



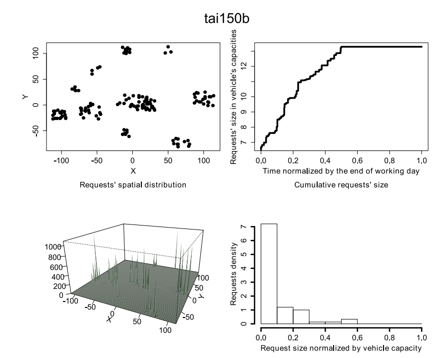
(a) Depiction of the **c50** benchmark set. **c50** is characterized by a uniform spatial distribution of requests that are relatively small and similar in size.



(b) Depiction of the **c120** benchmark set. **c120** is characterized by a clustered spatial distribution of requests that are relatively small and similar in size.

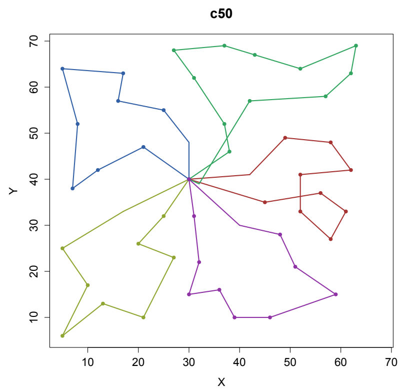


(c) Depiction of the **f134** benchmark set. **f134** is characterized by a semi-structured spatial distribution of requests with quite a high number of relatively small requests and several large requests appearing within the first 20% of a working day time.

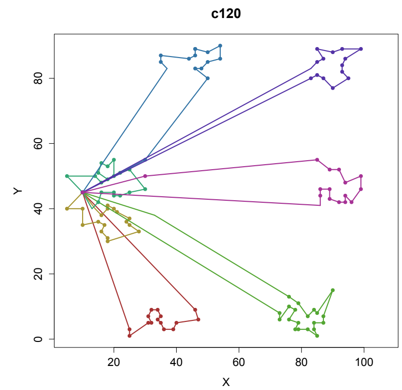


(d) Depiction of the **tai150b** benchmark set. **tai150b** is characterized by a clustered spatial distribution of requests with a relatively high number of (non-uniformly distributed) large requests. In addition, some of these large requests appear relatively late, i.e. after the first 25% of a working day time.

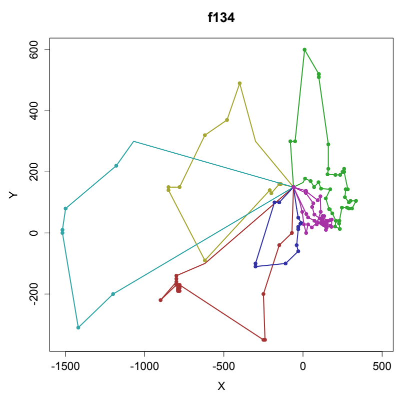
Figure B.9: Spatial and volume distribution of requests (left subplots in the subfigures), the plot of cumulative requests' size and the histogram of requests' sizes (right subplots in the subfigures).



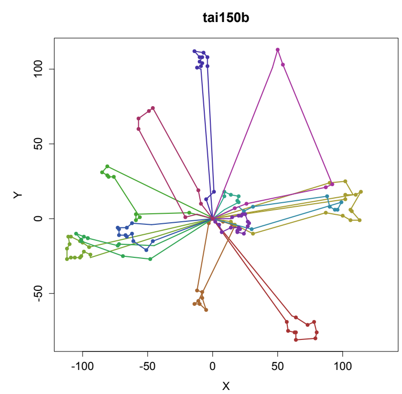
(a) Depiction of the result for the **c50** benchmark set.



(b) Depiction of the result for the **c120** benchmark set.



(c) Depiction of the result for the **f134** benchmark set.



(d) Depiction of the result for the **tai150b** benchmark set.

Figure B.10: Results obtained by 2MPSO for a selected benchmark instances.

Table B.8: Best result obtained for **c50**

Vehicle	Client	X	Y	Known	Scheduled
	Depot	30	40	0	351
1	38	45	35	0	42.14
1	50	56	37	0	68.32
1	34	61	33	0	135.48
1	30	58	27	0	157.19
1	9	52	33	42	181.44
1	16	52	41	80	204.44
1	21	62	42	128	229.49
1	29	58	48	0	251.70
1	2	49	49	4	275.75
1	11	42	41	51	301.38
2	47	25	32	0	70.86
2	4	20	26	12	93.67
2	17	27	23	86	145.47
2	42	21	10	0	174.78
2	19	13	13	104	198.33
2	40	5	6	0	223.96
2	41	10	17	0	251.04
2	13	5	25	63	275.47
2	18	17	33	100	304.90
3	32	38	46	0	36.33
3	1	37	52	1	57.41
3	8	31	62	34	84.07
3	26	27	68	0	118.74
3	31	37	69	0	143.79
3	28	43	67	0	165.11
3	3	52	64	9	189.60
3	36	63	69	0	216.68
3	35	62	63	0	237.76
3	20	57	58	116	259.83
3	22	42	57	138	289.87
3	46	32	39	0	325.46
4	6	21	47	16	125.48
4	14	12	42	79	150.77
4	25	7	38	0	172.18
4	24	8	52	0	201.21
4	43	5	64	0	228.58
4	7	17	63	21	255.62
4	23	16	57	157	276.70
4	48	25	55	0	300.92
4	27	30	48	0	324.53
5	12	31	32	58	95.81
5	37	32	22	0	120.86
5	44	30	15	0	143.14
5	15	36	16	80	164.23
5	45	39	10	0	185.93
5	33	46	10	0	207.93
5	39	59	15	0	236.86
5	10	51	21	44	261.86
5	49	48	28	0	284.48
5	5	40	30	12	307.72

Table B.9: Best result obtained for **c120**

Vehicle	Client	X	Y	Known	Time
	Depot	10	45	0	794
1	8	46	9	38	467.76
1	12	47	6	70	483.92
1	13	40	5	71	504.00
1	14	39	3	89	519.23
1	15	36	3	114	535.23
1	11	35	5	70	550.47
1	10	34	6	62	564.88
1	9	35	7	46	579.30
1	7	34	9	35	594.53
1	6	32	9	34	609.53
1	5	31	7	29	624.77
1	4	32	5	27	640.00
1	3	31	5	26	654.00
1	1	25	1	1	674.21
1	2	25	3	14	689.21
1	88	11	42	0	743.65
2	119	5	40	0	463.62
2	82	10	40	0	481.62
2	81	10	35	0	499.62
2	112	15	36	0	517.72
2	84	17	35	0	532.96
2	117	16	33	0	548.19
2	113	18	31	0	564.02
2	83	18	30	0	578.02
2	108	28	33	0	601.46
2	118	25	35	0	618.07
2	18	24	36	152	632.48
2	114	25	37	0	646.89
2	90	21	39	0	664.37
2	91	20	40	0	678.78
2	92	18	41	0	694.02
2	89	18	40	0	708.02
2	85	16	38	0	723.85
2	86	14	40	0	739.67
3	17	73	8	140	350.96
3	16	73	6	122	365.96
3	20	76	10	172	383.96
3	23	78	9	200	399.20
3	19	76	6	166	415.80
3	25	79	5	223	431.97
3	22	78	3	188	447.20
3	24	79	3	206	461.20
3	27	82	3	226	477.20
3	33	85	1	274	493.81
3	30	84	3	251	509.04
3	31	84	5	254	524.04
3	34	87	5	276	540.04
3	36	87	7	292	555.04
3	29	90	15	235	576.59
3	35	85	8	289	598.19
3	32	84	9	273	612.60
3	28	82	7	234	628.43
3	26	79	11	223	646.43
3	21	76	13	181	663.04
3	109	33	38	0	725.78
4	120	5	50	0	423.92
4	105	14	50	0	445.92
4	102	16	48	0	461.75
4	101	18	49	0	476.99
4	106	15	51	0	493.59
4	107	16	54	0	509.75
4	104	18	53	0	524.99
4	103	20	55	0	551.78
4	99	20	50	0	569.78
4	100	22	51	0	585.01
4	116	25	52	0	601.18
4	115	30	46	0	621.99
4	97	25	45	0	640.09
4	94	22	44	0	656.25
4	93	20	44	0	671.25
4	96	20	45	0	685.25
4	95	16	45	0	702.25
4	87	15	42	0	718.41
4	111	13	40	0	734.24
5	98	30	55	0	439.21
5	68	50	80	0	484.23
5	73	46	83	0	502.23
5	76	48	83	0	517.23
5	77	50	85	0	533.05
5	79	54	86	0	550.18
5	80	54	90	0	567.18
5	78	50	88	0	584.65
5	75	46	89	0	601.77
5	72	46	89	0	614.77
5	74	46	87	0	629.77
5	71	44	86	0	645.01
5	70	35	87	0	667.06
5	69	35	85	0	682.06
5	67	37	83	0	697.89
6	52	83	80	0	378.71
6	54	85	81	0	432.09
6	57	87	80	0	447.32
6	59	90	77	0	464.56
6	65	95	80	0	483.40
6	61	93	82	0	499.22
6	62	93	84	0	514.22
6	64	94	86	0	529.46
6	66	99	89	0	548.29
6	63	93	89	0	567.29
6	60	90	88	0	583.45
6	56	85	89	0	601.55
6	58	87	86	0	618.16
6	55	85	85	0	633.39
6	53	83	83	0	649.22
7	110	30	50	0	377.92
7	40	85	55	324	446.14
7	43	89	52	339	464.14
7	45	92	52	356	480.14
7	48	94	48	377	497.61
7	51	99	50	0	516.00
7	50	99	46	396	533.00
7	49	96	42	383	551.00
7	47	94	44	361	566.83
7	46	94	42	358	581.83
7	44	92	42	346	596.83
7	41	89	43	330	612.99
7	42	89	46	330	628.99
7	39	86	46	324	644.99
7	38	86	44	324	659.99
7	37	86	41	304	675.99

Table B.10: Best result obtained for **f134**

Vehicle	Client	X	Y	Known	Time
	Depot	-60	150	0	11741
1	78	-70	0	0	7488.46
1	133	-150	-40	0	7590.90
1	68	-250	-200	4650	7792.58
1	70	-240	-350	5063	7955.91
1	69	-250	-350	4975	7978.91
1	112	-780	-170	0	8551.65
1	125	-780	-175	0	9997.85
1	111	-780	-180	0	10015.85
1	110	-780	-190	0	10038.85
1	122	-790	-190	0	10061.85
1	123	-790	-185	0	10079.85
1	124	-790	-180	0	10097.85
1	126	-790	-170	0	10120.85
1	127	-800	-170	0	10143.85
1	121	-900	-220	0	10268.65
1	128	-800	-160	0	10398.27
1	129	-800	-150	0	10421.27
1	113	-800	-140	0	10444.27
1	81	-620	-100	0	10641.66
2	46	-140	160	3073	8886.37
2	118	-150	160	0	8909.37
2	17	-200	130	1171	8980.68
2	18	-210	140	1234	9007.82
2	132	-620	-90	0	9490.93
2	116	-850	140	0	9829.20
2	131	-850	150	0	9852.20
2	117	-780	150	0	9935.20
2	119	-620	320	0	10181.65
2	130	-480	370	0	10343.31
2	65	-400	490	4479	10500.53
2	19	-300	300	1278	10728.24
3	82	-80	300	0	8957.08
3	20	-50	300	1278	9000.08
3	83	10	600	0	9319.02
3	85	100	520	0	9452.43
3	84	100	520	0	9465.43
3	86	100	510	0	9488.43
3	87	160	290	0	9729.47
3	89	160	210	0	9822.47
3	90	155	192	0	9854.15
3	16	205	190	1040	9917.19
3	13	235	190	909	9960.19
3	15	250	200	1039	9991.22
3	88	260	210	0	10018.36
3	14	260	200	953	10041.36
3	11	283	143	672	10115.83
3	12	270	143	716	10141.83
3	10	265	117	587	10181.30
3	9	290	100	541	10224.54
3	8	300	105	453	10248.72
3	7	335	105	389	10296.72
3	6	310	80	345	10345.07
3	5	290	80	282	10378.07
3	4	278	83	259	10403.44
3	2	246	83	111	10448.44
3	42	230	40	2898	10507.32
3	41	228	31	2702	10529.54
3	3	233	13	130	10561.22
3	40	203	21	2702	10605.27
3	44	208	40	3030	10637.92
3	43	208	40	3006	10650.92
3	45	185	64	3049	10697.16
3	94	169	77	0	10730.77
3	93	165	78	0	10747.90
3	29	144	113	1969	10801.71
3	92	172	143	0	10855.75
3	28	114	145	1969	10926.78
3	27	98	166	1967	10966.19
3	26	70	150	1819	11011.43
3	25	48	170	1754	11054.17
3	21	13	178	1430	11103.07
3	91	0	165	0	11134.45
4	120	-1200	-200	0	8824.17
4	109	-1420	-310	0	9083.14
4	108	-1520	0	0	9421.87
4	107	-1520	10	0	9444.87
4	106	-1500	80	0	9530.67
4	114	-1180	220	0	9892.95
4	115	-1070	300	0	10041.97
5	66	-150	100	4543	10082.81
5	71	-180	100	5319	10125.81
5	33	-300	-100	2164	10372.04
5	80	-300	-110	0	10395.04
5	67	-110	-100	4648	10598.31
5	79	-30	-60	0	10700.75
5	63	-40	-40	4326	10736.11
5	64	-30	12	4327	10802.06
5	77	-30	20	5817	10823.06
5	76	-17	30	5751	10852.46
5	134	-10	32	0	10872.74
5	74	-30	50	5603	10912.65
5	73	-40	80	5538	10957.27
6	75	21	62	5729	9512.40
6	1	32	51	25	9540.96
6	62	72	40	4281	9595.44
6	50	87	28	3547	9627.65
6	51	90	33	3632	9646.49
6	53	112	33	3719	9681.49
6	102	118	30	0	9701.19
6	103	120	40	0	9724.39
6	104	128	36	0	9746.34
6	101	130	26	0	9769.53
6	35	145	10	2399	9804.47
6	36	150	18	2488	9826.90
6	37	172	24	2509	9862.70
6	95	180	20	0	9884.65
6	39	182	44	2598	9921.73
6	38	172	42	2550	9944.93
6	96	162	40	0	9968.13
6	97	150	40	0	9993.13
6	98	150	30	0	10016.13
6	99	148	24	0	10035.45
6	100	145	30	0	10055.16
6	105	134	55	0	10095.47
6	57	123	55	4043	10119.47
6	56	123	47	3874	10140.47
6	55	115	46	3828	10161.53
6	54	108	47	3762	10181.61
6	61	72	60	4194	10232.88
6	60	58	85	4155	10274.53
6	59	65	97	4066	10301.43
6	23	18	131	1495	10372.43
6	22	18	138	1492	10544.63
6	24	20	136	1666	10560.46
6	31	93	107	2098	10652.01
6	30	110	120	2096	10686.41
6	58	112	69	4066	10750.45
6	52	90	35	3700	10803.95
6	49	56	18	3354	10854.96
6	48	32	28	3265	10893.96
6	34	20	0	2273	10937.42
6	32	6	28	2142	10981.73
6	47	-5	69	3203	11037.18
6	72	-20	100	5386	11084.62

Table B.11: Best result obtained for **tai150b**

Vehicle	Client	X	Y	Known	Time
	Depot	0	0	0	988
1	8	57	-69	45	484.70
1	1	58	-75	3	517.78
1	7	63	-76	34	549.88
1	9	64	-76	51	577.88
1	10	64	-81	74	609.88
1	3	79	-80	7	651.91
1	2	80	-76	5	683.04
1	11	78	-69	74	717.32
1	6	73	-71	31	749.70
1	4	64	-66	18	787.00
1	5	61	-65	23	817.16
2	148	-3	-3	0	275.94
2	84	-12	-48	0	348.83
2	86	-8	-49	0	379.96
2	82	-8	-53	0	500.30
2	87	-5	-61	0	535.84
2	85	-10	-57	0	569.25
2	88	-14	-57	0	697.90
2	83	-11	-55	0	728.51
2	16	17	-5	100	812.81
2	120	15	-4	0	842.05
2	22	8	-2	131	876.33
2	28	7	-2	149	904.33
3	127	14	-2	0	63.54
3	136	17	-4	0	94.15
3	137	31	-10	0	136.38
3	106	87	4	0	221.10
3	102	100	2	0	261.26
3	96	106	-1	0	294.96
3	99	113	-1	0	328.96
3	95	107	5	0	364.45
3	89	106	6	0	392.86
3	104	114	18	0	434.29
3	107	110	16	0	465.76
3	97	102	16	0	500.76
3	92	102	13	0	530.76
3	93	90	24	0	753.73
3	108	102	25	0	792.77
3	103	108	16	0	830.59
4	33	-95	-19	207	417.98
4	47	-100	-16	298	450.81
4	35	-102	-15	211	480.05
4	49	-109	-12	300	514.66
4	48	-111	-12	300	543.66
4	37	-110	-17	218	575.76
4	32	-112	-20	180	606.37
4	34	-112	-27	207	640.37
4	41	-109	-26	247	670.53
4	40	-106	-26	244	700.53
4	42	-102	-26	258	731.53
4	29	-101	-25	154	759.95
4	31	-99	-22	174	790.55
4	45	-94	-24	282	822.94
4	30	-94	-26	165	851.94
5	149	-18	4	0	364.24
5	52	-59	3	320	432.25
5	56	-59	-1	338	463.25
5	62	-56	1	371	574.01
5	81	-76	28	0	634.61
5	77	-80	28	0	697.90
5	80	-81	29	0	726.31
5	79	-85	31	0	757.79
5	78	-81	35	0	790.44
5	145	-1	1	0	904.37
5	26	-2	0	142	934.78
6	51	-48	-27	311	425.57
6	61	-69	-25	366	473.67
6	38	-101	-15	222	579.23
6	43	-101	-14	271	607.23
6	39	-105	-10	227	639.88
6	44	-99	-12	280	673.21
6	46	-96	-13	293	703.37
6	36	-96	-13	213	730.37
6	68	-73	-18	453	780.91
6	59	-72	-17	362	809.32
6	66	-45	-18	426	863.34
7	23	9	15	133	709.09
7	15	9	18	96	739.09
7	25	14	16	142	771.48
7	19	20	15	109	804.56
7	21	19	12	123	834.72
7	27	20	11	144	863.14
7	20	9	6	114	902.22
8	121	30	-7	0	277.81
8	105	99	11	0	376.12
8	90	96	6	0	650.33
8	94	94	6	0	679.33
8	98	90	8	0	710.80
8	100	88	15	0	745.08
8	122	31	8	0	829.51
8	24	24	5	136	864.13
8	18	20	8	102	896.13
9	65	-46	-15	415	542.38
9	67	-51	-21	447	577.19
9	69	-61	-15	455	615.86
9	54	-62	-10	320	647.96
9	50	-66	-12	302	679.43
9	60	-67	-11	365	707.84
9	55	-72	-11	333	739.84
9	64	-72	-7	407	770.84
9	57	-73	-6	353	799.26
9	63	-66	-6	402	833.26
9	58	-62	-3	355	865.26
9	53	-47	-4	320	907.29
10	140	-5	13	0	260.93
10	141	1	18	0	295.74
10	116	-4	102	0	406.89
10	112	-4	108	0	477.60
10	115	-7	111	0	508.84
10	118	-14	112	0	542.91
10	111	-10	108	0	575.57
10	109	-8	108	0	671.20
10	119	-10	105	0	701.81
10	113	-8	104	0	731.04
10	110	-9	102	0	760.28
10	117	-12	101	0	790.44
10	114	-9	100	0	820.60
11	142	2	-2	0	225.13
11	144	5	-4	0	255.73
11	14	7	-9	96	288.12
11	17	14	-6	102	322.73
11	131	20	-6	0	477.60
11	128	21	-7	0	506.01
11	125	19	-8	0	535.25
11	126	19	-9	0	563.25
11	129	24	-10	0	595.35
11	133	27	-5	0	650.33
11	139	28	-3	0	679.57
11	135	27	-2	0	707.98
11	134	25	3	0	740.37
11	132	23	3	0	769.37
11	138	22	2	0	797.78
11	12	20	2	85	826.78
11	123	15	1	0	858.88
11	13	14	0	87	887.29
12	124	17	7	0	92.48
12	130	26	10	0	128.97
12	101	87	21	0	217.96
12	91	92	23	0	250.34
12	76	54	103	0	733.07
12	75	50	113	0	770.84
12	74	46	101	0	810.49
13	143	-9	10	0	556.85
13	146	-11	19	0	593.07
13	71	-46	74	486	685.27
13	73	-49	72	491	715.87
13	72	-57	67	487	752.30
13	70	-57	60	460	786.30
13	150	-21	1	0	882.42
13	147	-12	3	0	918.64
166	101	87	21	0	217.96
166	91	92	23	0	250.34
166	76	54	103	0	733.07
166	75	50	113	0	770.84
166	74	46	101	0	810.49
170	143	-9	10	0	556.85
170	146	-11	19	0	593.07
170	71	-46	74	486	685.27
170	73	-49	72	491	715.87
170	72	-57	67	487	752.30
170	70	-57	60	460	786.30
170	150	-21	1	0	882.42
170	147	-12	3	0	918.64

References

[1] H. N. Psaraftis, Dynamic vehicle routing: Status and prospects, Annals of Operations Research 61 (1995) 143–164.

[2] G. B. Dantzing, J. H. Ramser, The Truck Dispatching Problem, Management Science 6 (1) (1959) 80–91. `doi:10.1287/mnsc.6.1.80`.
URL `http://www.jstor.org/stable/2627477`

[3] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, A review of dynamic vehicle routing problems., European Journal of Operational Research 225 (1) (2013) 1–11.
URL `http://dblp.uni-trier.de/db/journals/eor/eor225.html#PillacGGM13`

[4] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Applied Soft Computing 13 (10) (2013) 4023–4037.

[5] M. R. Khouadjia, E.-G. Talbi, L. Jourdan, B. Sarasola, E. Alba, Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems, Journal of Supercomputing 63 (3) (2013) 836–853.

[6] M. Okulewicz, J. Mańdziuk, Two-Phase Multi-Swarm PSO and the Dynamic Vehicle Routing Problem, in: 2nd IEEE Symposium on Computational Intelligence for Human-like Intelligence, IEEE, Orlando, Fl, USA, 2014, pp. 86–93. `doi:10.1109/CIHLI.2014.7013391`.

[7] T. Blackwell, Particle swarm optimization in dynamic environments, Evolutionary Computation in Dynamic and Uncertain Environments 51 (2007) 29–49.

[8] P. Kilby, P. Prosser, P. Shaw, Dynamic VRPs: A Study of Scenarios (1998).
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.6748`

[9] C. Lin, K. L. Choy, G. T. S. Ho, H. Y. Lam, G. K. H. Pang, K. S. Chin, A decision support system for optimizing dynamic courier routing operations, Expert Systems with Applications 41 (15) (2014) 6917–6933. `doi:10.1016/j.eswa.2014.04.036`.

[10] R. Montemanni, L. M. Gambardella, A. Rizzoli, A. Donati, A new algorithm for a dynamic vehicle routing problem based on ant colony system, Journal of Combinatorial Optimization 10 (2005) 327–343.

[11] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, E.-G. Talbi, A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests, Applied Soft Computing 12 (4) (2012) 1426–1439. `doi:10.1016/j.asoc.2011.10.023`.

[12] M. R. Khouadjia, E. Alba, L. Jourdan, E.-G. Talbi, Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem, in: Swarm Intelligence, Vol. 6234 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, 2010, pp. 227–238. doi:10.1007/978-3-642-15461-4_20.

[13] F. T. Hanshar, B. M. Ombuki-Berman, Dynamic vehicle routing using genetic algorithms, Applied Intelligence 27 (1) (2007) 89–99. doi:10.1007/s10489-006-0033-z.

[14] M. Okulewicz, J. Mańdziuk, Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7895 (2) (2013) 547–558. doi:10.1007/978-3-642-38610-7_50.

[15] P. Garrido, M. C. Riff, DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic, Journal of Heuristics 16 (6) (2010) 795–834.

[16] M. J. Elhassania, B. Jaouad, E. A. Ahmed, A new hybrid algorithm to solve the vehicle routing problem in the dynamic environment, International Journal of Soft Computing 8 (5) (2013) 327–334.

[17] J. Mańdziuk, A. Żychowski, A memetic approach to vehicle routing problem with dynamic requests, Applied Soft Computing 48 (2016) 522–534. doi:http://dx.doi.org/10.1016/j.asoc.2016.06.032.

[18] T. C. Du, E. Y. Li, D. Chou, Dynamic vehicle routing for online B2C delivery, Omega 33 (1) (2005) 33–45. doi:10.1016/j.omega.2004.03.005.

[19] P. Garrido, C. Castro, Stable solving of {CVRPs} using hyperheuristics, Proceedings of the 11th Annual conference on Genetic and evolutionary computation (2009) 255–262doi:10.1145/1569901.1569938.

[20] P. Garrido, C. Castro, A Flexible and Adaptive Hyper-heuristic Approach for (Dynamic) Capacitated Vehicle Routing Problems, Fundamenta Informaticae 119 (1) (2012) 29–60. doi:10.3233/FI-2012-726.

[21] M. Elhassania, B. Jaouad, E. A. Ahmed, Solving the dynamic Vehicle Routing Problem using genetic algorithms, in: Logistics and Operations Management (GOL), 2014 International Conference on, IEEE, 2014, pp. 62–69.

[22] M. Okulewicz, J. Mańdziuk, Particle Swarm Optimization hyper-heuristic for the Dynamic Vehicle Routing Problem, in: 7th BIOMA Conference, 2016, pp. 215–227. doi:10.13140/RG.2.2.27509.58082.

[23] J. Kennedy, R. Eberhart, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks. IV (1995) 1942–1948.

[24] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, Proceedings of Evolutionary Programming VII (EP98) (1998) 591–600.

[25] Y. Shi, R. C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation (1998) 69–73.

[26] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317–325.

[27] M. Clerc, Standard PSO 2007 and 2011 (2012).
URL `http://www.particleswarm.info/`

[28] C. K. Monson, K. D. Seppi, Exposing origin-seeking bias in PSO, in: Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO '05, ACM, New York, NY, USA, 2005, pp. 241–248. `doi:10.1145/1068009.1068045`.

[29] W. M. Spears, D. T. Green, D. F. Spears, Biases in Particle Swarm Optimization, International Journal of Swarm Intelligence Research 1(2) (2010) 34–57.

[30] T. J. Ai, V. Kachitvichyanukul, A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery, Computers and Operations Research 36 (5) (2009) 1693–1702. `arXiv:arXiv:1011.1669v3`, `doi:10.1016/j.cor.2008.04.003`.

[31] S. Wang, L. Wang, H. Yuan, M. Ge, B. Niu, W. Pang, Y. Liu, Study on Multi-Depots Vehicle Scheduling Problem and Its Two-Phase Particle Swarm Optimization, no. 2006, 2009, pp. 748–756. `doi:10.1007/978-3-642-04020-7_80`.
URL `http://link.springer.com/10.1007/978-3-642-04020-7_80`

[32] Y. Marinakis, G.-R. Iordanidou, M. Marinaki, Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands, Applied Soft Computing Journal 13 (4) (2013) 1693–1704. `doi:10.1016/j.asoc.2013.01.007`.

[33] T. J. Ai, V. Kachitvichyanukul, Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem, Computers & Industrial Engineering 56 (1) (2009) 380–387. `doi:10.1016/j.cie.2008.06.012`.

[34] M. Okulewicz, Source code of the Two-Phase Multiswarm Particle Swarm Optimizer for Dynamic Vehicle Routing Problem (2016).
URL `https://sourceforge.net/projects/continuous-dvrp/`

[35] J. B. Kruskal, On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem, Proceedings of the American Mathematical Society (1959) 48–50.

[36] G. A. Croes, A method for solving traveling salesman problems, Operations Res. 6 (1958) 791–812.

[37] N. Christofides, J. E. Beasley, The period routing problem, Networks 14 (2) (1984) 237–256. doi:10.1002/net.3230140205.

[38] M. L. Fisher, R. Jaikumar, A generalized assignment heuristic for vehicle routing, Networks 11 (2) (1981) 109–124. doi:10.1002/net.3230110205. URL http://dx.doi.org/10.1002/net.3230110205

[39] É. D. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (8) (1993) 661–673. doi:10.1002/net.3230230804. URL http://onlinelibrary.wiley.com/doi/10.1002/net.3230230804/abstract

[40] J. Mańdziuk, S. Zadrożny, K. Walędzik, M. Okulewicz, M. Świechowski, Adaptive metaheuristic methods in dynamically changing environments (2015).
URL http://www.mini.pw.edu.pl/~mandziuk/dynamic/