A Two-Phase Multi-Swarm Algorithm for Solving Dynamic Vehicle Routing Problem

Michał Okulewicz, Jacek Mańdziuk, *Senior Member, IEEE*

***Abstract*—This paper describes a method of solving Dynamic Vehicle Routing Problem (DVRP) with Particle Swarm Optimization. In the particular embodiment of DVRP considered in this paper new requests may arrive during the whole working day. Hence, the working day is divided into a certain number of time slots in which a static Vehicle Routing Problem is solved. The solution method works in two phases: the clustering phase and the route optimization phase. Each of them is addressed by a separate multi-swarm PSO instance.**

Three forms of the fitness functions used for cost assessment of the tours and two problem encodings are proposed and experimentally verified.

The approach is tested on a well established set of benchmark problems and produces more efficient solutions (on average from 1.5% for identical number of fitness function evaluations (FFE) up to 5.1% with increased number of FFE) than the best metaheuristic results reported in the literature.

For the comparable number of FFE our method yielded 11 (out of 21 tested benchmark instances) new best literature results. Additionally, with further increase of the number of FFE it was capable of improving the best known solutions in 17 problems (out of these 21).

I. Introduction

THE static version of the Vehicle Routing Problem (VRP), introduced in the literature under the name *The truck dispatching problem*, was initially proposed by Dantzig and Ramser [1] and proved to be NP-Hard by Lenstra and Kan [2].

Dynamic Vehicle Routing Problem (DVRP) is a generalization of the VRP and in the literature, there are several versions of the DVRP considered, which differ mainly by the way a dynamic/stochastic factor is introduced. The version discussed in this paper is the most common description of the problem [3], sometimes referred to as the Vehicle Routing Problem with Dynamic Requests (VRP+DR) [4]. In the DVRP (VRP+DR) the goal is to find a minimal route for a fleet of vehicles within a given time and capacity constrains without having full knowledge of the number, location and size of the set of requests at the start of the working day.

Generally speaking, in each point in time the DVRP may be looked at as a combination of the two NP-Complete problems: the Bin Packing Problem (BPP) for assigning the requests to vehicles (clustering the set of requests) and the Traveling Salesman Problem (TSP) for finding an optimal route for each vehicle.

Division of the solution method into assignment and permutation phases was initially proposed in heuristic algorithms used for solving the static VRP by Fischer and Jaikumar [5], Ryan et al. [6] and Taillard [7]. Such a combination of BPP and TSP may be effectively solved by both approximate methods [8], [9], [10] and metaheuristic algorithms, e.g. [11], [12], [13], [14]. In this paper, the latter approach is adopted and the Particle Swarm Optimization (PSO) algorithm is used to solve both of these sub-problems. More precisely, the proposed method - *Two-Phase Multi-Swarm PSO* (2MPSO) maintains several swarms which simultaneously search for the optimal solution. Utilization of a few swarms, enhanced by local optimization of the tours and application of a new problem encoding led to visible improvement of results compared to our initial (single-swarm) approach [15].

Furthermore, when comparing the results with the state-of-the-art solutions, for the *cut-off time* (see subsection II-C for its description and meaning) set in the middle of a day (which is the most common choice in the literature), the proposed approach yields new best literature results in the case of 17 out of 21 benchmark problems considered.

This paper significantly extends our previous work [16] mainly by providing a more thorough in-depth explanation of the 2MPSO algorithm (in particular the problem encoding in both phases) and by presenting certain limitations of the clustering approach used in the first phase. Additionally, the proposed method is compared in detail with its main competitor - the state-of-the-art version of the MEMSO [17] method, in terms of problem encodings and fitness functions used.

The proposed multi-swarm method is also evaluated against a single-swarm implementation with one large swarm of the respective size. The advantages and weaknesses of both implementations are presented and discussed.

Furthermore, an analysis of the contribution of several auxiliary heuristics employed in our Two-Phase PSO method is performed leading to a conclusion that the core strength of the method should be mainly attributed to the PSO-based clustering phase (the first phase of the method).

It is also argued in this paper that dividing the working day into greater number of time slices than that proposed in the literature [8] (and also used in our initial 2MPSO presentation [16]) proved beneficial as it led to revealing new, previously unknown best results for 13 benchmark problems.

Unlike our previous works [15], [16] which only formally sketched the DVRP definition, this paper includes a detailed discussion on a general framework used for solving this problem as a series of static instances. Moreover, the role of the related steering parameters: the *cut-off time*, the *advanced commitment time* and the *number of time slices*, as well as the pertinence of their proper selection are discussed in detail.

The paper is organized as follows. In the next section the

Michał Okulewicz and Jacek Mańdziuk are with the Faculty of Mathematics and Information Science of Warsaw University of Technology, Warsaw, Poland (email: M.Okulewicz, J.Mandziuk@mini.pw.edu.pl).

definitions of the PSO algorithm and the DVRP are given. In section III several metaheuristic approaches to solving DVRP, which will be used for comparison with our method, are presented. The following section describes our 2MPSO algorithm and compares it with previous PSO-based approaches. Subsequently, section V discusses the 2MPSO parameter setup and introduces the set of benchmark problems used to verify the efficacy of the method. Section VI presents the results of experimental evaluation of the 2MPSO in comparison with the competitive methods described in section III. The last section summarizes the outcomes, provides conclusions and points directions for possible future extensions of this work.

II. Definitions of PSO and DVRP

In this section the two major components of the presented research, namely the Particle Swarm Optimization and the Dynamic Vehicle Routing Problem are introduced.

A. Particle Swarm Optimization

PSO is an iterative global optimization metaheuristic method proposed in 1995 by Kennedy and Eberhart [18] and further studied and developed by many other researchers, e.g., [19],[20],[21]. In short, PSO utilizes the idea of swarm intelligence to solve hard optimization tasks. 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 time step t, each particle i updates its position x_t^i and velocity v_t^i according to the following formulas:

Position update: The position is updated according to the following equation:

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

Velocity update: In our implementation of PSO (based on [22] and [19]) velocity v_t^i of particle i is updated according to the following rule:

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

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.

In our study we use the Standard Particle Swarm Optimization 2007 (SPSO-07) [22] with random star neighborhood topology, in which, for each particle, we randomly assign its neighbors, each of them independently, with a given probability[1]. Please note, that we are using SPSO-07 instead of the newer SPSO-11 [22] 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 [23], [24]. Since in our method the search space center is defined at the current-best particle position, the above mentioned bias increases the probability of finding the particles in the nearby of this best location.

B. Dynamic Vehicle Routing Problem

In the version of DVRP discussed in this article one considers:

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

The fleet V is homogeneous, i.e. vehicles have identical $capacity \in \mathbb{R}$ and the same $speed$[2] $\in \mathbb{R}$. The cargo is taken from one of the k depots[3]. Each depot $d_j \in D, j = 1, \ldots, k$ has:

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

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

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

- $location_l \in \mathbb{R}^2$,
- $time_l \in \mathbb{R}$, which is a point in time when their request becomes available $(start \leq time_l \leq end)$,
- $unld_l \in \mathbb{R}$, which is the time required to unload the cargo,
- $size_l \in \mathbb{R}$ - which is the size of the request $(size_l \leq capacity)$.

A travel distance $\rho(i, j)$ is the Euclidean distance between $location_i$ and $location_j$ on the $\mathbb{R}^2$ plane, $i, j = 1, \ldots, m+k$.

For each vehicle v_i, the $r_i = (i_1, i_2, \ldots, i_{p(i)})$ is a permutation of indexes of requests assigned to the ith vehicle and depots to be visited by the vehicle, which 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_{i_j} is the time of arrival to the jth request of the ith vehicle. arv_{i_j} is induced by the permutation r_i, the time when requests become available - see eqs. (4) and (5) and the time arv_{i_1} at which vehicle leaves the depot.

[1] 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.

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

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

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

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:

$$COST(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{p(i)} \rho(i_j, i_{j-1}) \tag{3}$$

under the following constraints (4) - (8).

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

$$\begin{aligned} \forall_{i \in \{1,2,\ldots,n\}} \forall_{j \in r_i / \{i_1\}} \quad & arv_{i_j} \\ & \geq time_{i_j} + \rho(i_j, i_{j-1}) \end{aligned} \tag{4}$$

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

The vehicle cannot arrive at $location_{i_j}$ before serving the request $r_{i_{j-1}}$ and traveling to the next location:

$$\begin{aligned} \forall_{i \in \{1,2,\ldots,n\}} \forall_{j \in r_i / \{i_1\}} \quad & arv_{i_j} \\ \geq arv_{i_{j-1}} + unld_{i_{j-1}} + \rho(i_j, i_{j-1}) & \end{aligned} \tag{5}$$

All vehicles must return to the depot before its closing:

$$\forall_{i \in \{1,2,\ldots,n\}} \quad arv_{i_{p(i)}} \leq end_{i_{p(i)}} \tag{6}$$

Recall that index $i_{p(i)}$ (the last index in route 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:

$$\begin{aligned} \forall_{i \in \{1,2,\ldots,n\}} \forall_{j_1, j_2 \in r_i, j_1 < j_2, i_{j_1} \leq k} : \quad & \sum_{l=j_1}^{j_2} size_{i_l} \\ \leq capacity \cdot |\{l' : i_{l'} \leq k \wedge l' \in [j_1, j_2 - 1]\}| & \end{aligned} \tag{7}$$

Each client must be assigned to exactly one vehicle:

$$\forall_{l \in \{1,2,\ldots,m\}} \exists!_{i \in \{1,2,\ldots n\}} \quad (k + l) \in r_i \tag{8}$$

C. Solution framework

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 actual problem instance and sends it to the optimization module and, afterwards, uses the solution found to commit vehicles. Such a framework is depicted in Fig. 2. An example of a technical description of such information technology system could be found in [25].

The event scheduler maintains the three following parameters, which in some sense define the "degree of dynamism" of a given problem instance:

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

The *cut-off* time (T_{co}) parameter, in real business situations, could be interpreted as a time threshold for not accepting any new requests that arrive afterwards 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 [14], [4], [26], [17], [13], [15], the requests that arrive after the T_{co} are treated as being known at the beginning of the *current* day, i.e. they actually compose the initial problem instance. In all tests, for the sake of comparability of 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 25, claiming the optimal trade-off between the quality of solutions and computation time. In the case of our algorithm we observed that it is beneficial to increase n_{ts} to 50 and, at the same time, decrease twice (from $100 + 25$ to $50 + 12$) the number of PSO iterations per time slice (respectively for the first and the second phase of the algorithm), thus maintaining the total number of fitness function evaluations (FFE) at the same level (equal to 10^6) for the sake of comparability with other literature results. 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.

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 finally approved and "frozen", i.e. the vehicle is dispatched to serve the respective requests. In other words, any requests whose respective vehicles are scheduled for the departure not later than **at the end of the current time slice span plus T_{ac} time** need to be dispatched

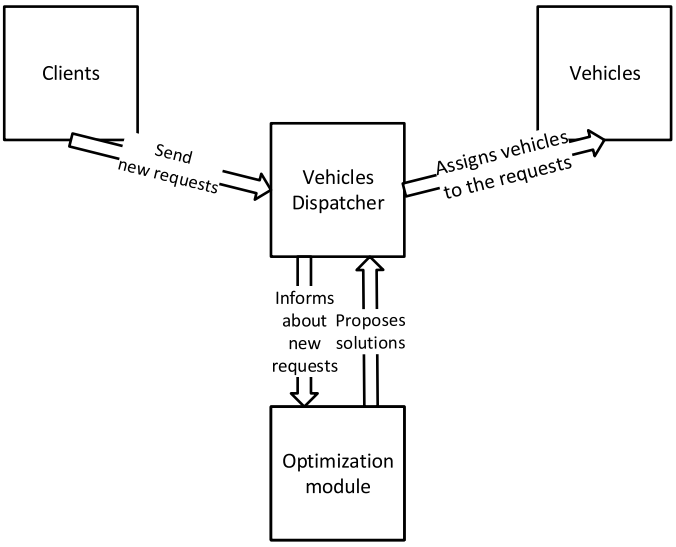


Fig. 2. General solution framework.

{The initial problem = all the requests received after the *cut-off* time}
for all $time\_slices$ **do**
Solve the current VRP instance
Select the best solution found {for this time slice}
Commit orders to respective vehicles {with processing time within the next $\frac{T}{n_{ts}} + T_{ac}$ seconds}
Include requests received during this time slice in the next static instance (next time slice)
Send back to the depot any vehicle which might otherwise be late at the depot closing
Send back to the depot any vehicle that used its capacity
end for

Fig. 1. An overview of DVRP processing schema when transformed into a sequence of static VRP instances.

immediately. In effect, a condition defined by eq. (6) is, in practice, changed into

$$\forall_{i\in\{1,2,\ldots,n\}} \quad arv_{i_{p(i)}} + T_{ac} \leq end_{i_{p(i)}} \tag{9}$$

We have observed that appropriate choice of this parameter 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. In our tests we set T_{ac} equal to 2 time slices:

$$T_{ac} = 2\frac{end - start}{n_{ts}} \tag{10}$$

A general algorithm for solving the DVRP problem as a sequence of static VRP instances is presented in Fig. 1.

III. Solving the DVRP

On a general note there are two major approaches to solving dynamic transportation (optimization) problems. In the first one the optimization algorithm is run whenever there is a change in the problem instance. 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 the following subsections we briefly review various problem encodings proposed in the literature used by the most effective Computational Intelligence (CI) approaches. All of them take advantage of a natural, graph-based, topological representation. In order to compare different ways of coding the problem and point differences in the methods' internal operators a common example of particular vehicles' assignment, with three vehicles and ten requests, will be used.

A. Discrete one-phase approaches

Initial metaheuristic methods devoted to solving the DVRP were largely one-phase methods, that simultaneously optimized the assignment of requests and the final lengths of the vehicles' tours. In particular this paradigm was implemented with the help of Ant Colony Systems (ACS) [14], Genetic Algorithms (GA) [13] and Tabu Search [13].

B. Ant Colony System

Application of Ant Colony System to DVRP [14] takes a direct advantage of graph representation of the problem. Similarly to the case of solving the TSP [27], when solving DVRP, ACS maintains a colony of ants, each of which is devoted to solving the whole problem and the best overall solution is finally selected. Ants communicate via pheromone trail which is deposited on their way (on the edges of a graph) with the amount proportional to the quality of obtained solution. Whenever the total load would exceed the truck's capacity an ant returns to the depot (initial vertex). This way each solution found by an ant is composed of a set of cycles of loads not greater than the truck's capacity.

Once a solution is found, it is further optimized by a greedy local search procedure, in which, iteratively, for each client's request all its possible placements in the assigned tour and in the other tours are tried. Whenever such a replacement leads to a solution improvement it is accepted.

C. Genetic Algorithm

Application of GA to solving DVRP was proposed by Hanshar and Ombuki-Berman [13]. The main features of this approach can be summarized as follows:

- Each genotype is a sequence of requests' identifiers (represented by positive integers) and vehicles' identifiers (represented by negative integers). All requests which appear after a given vehicle's id. until the next vehicle's id. define the current route of that vehicle.
- The cross-over operator creates two children from each pair of parent genotypes. In each parent chromosome one of the vehicle's routes is randomly chosen and all requests assigned to that route are removed from the other parent chromosome. Next, the removed requests are inserted in a greedy manner, i.e. in the places which lead to the shortest overall sum of vehicles' routes (all possible insertions in all routes are verified).
- The mutation operator randomly selects two points in a chromosome and reverses the order of all elements (clients' requests) located between these points.
- The frozen parts of the routes (the requests which are already ultimately assigned to vehicles) are not altered by the GA operators.
- 4 best specimen (1% of the population) are always selected unaltered by mutation and cross-over operators.

D. Tabu Search

Application of Tabu Search (TS) metaheuristic to solving DVRP was also proposed by Hanshar and Ombuki-Berman in the same paper [13]. The same problem representation as in the case of the GA method is used. The solution space is searched by choosing the best neighbor of the current solution except for those being on a tabu list and therefore unavailable for choosing. The neighborhood of the current solution is randomly selected (with probabilities equal to 0.6 and 0.4, respectively) to be one of the two following choices:

- 100 random solutions created by a mutation operator (inversion) described in the previous subsection, or
- the set of solutions created by the Osman and Christofides [28] $\lambda-$interchange operation. In short, this operation has two parameters: k and l and for each pair of vehicles used in the current solution (say veh_i and veh_j) it considers k requests assigned to the first vehicle and l requests assigned to the other vehicle and inserts them into the other vehicle's route in all possible combinations of their new positions, i.e. k requests are moved from veh_i to veh_j and l requests are moved from veh_j to veh_i and placed in all possible combinations of respectively k positions in veh_j route and l positions in veh_i route).

The implementation details of both GA and TS approaches can be found in [13].

Due to their one-phase characteristics all three of the above-mentioned population-based approaches utilize the cost function that directly optimizes the total length of the final tours:

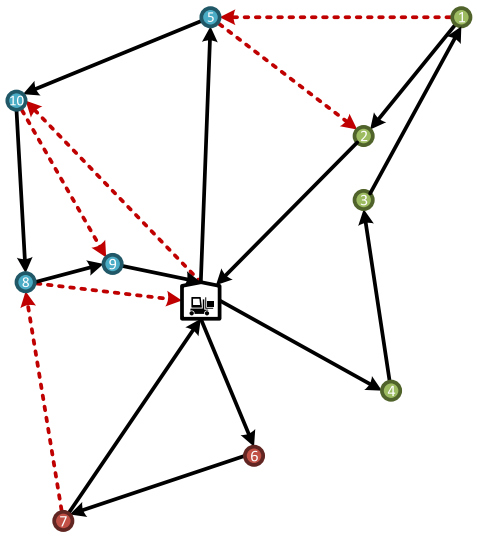
$$COST_3(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{|r_i|} \rho(i_j, i_{j-1}) \qquad (11)$$

E. DAPSO/MAPSO/MEMSO Algorithms

A Dynamic Adapted PSO (DAPSO) and its multi-swarm equivalents: Multiswarm Adaptive Memory PSO (MAPSO) and Multi-Environmental Multi-Swarm Optimizer (MEMSO) proposed by Khouadjia et al. [4], [26], [17] require modification of the PSO algorithm allowing their proper application in a discrete search space of the DVRP. The main characteristic features of the Khouadjia et al.'s approach are listed below:

- Each particle represents an integer vector containing identifiers of vehicles respectively assigned to requests to be served (see section IV-F).
- The route for each of the vehicles is constructed by a greedy insertion algorithm and improved by the 2-OPT method [29].
- All particles are propagated to the next time slice (and subsequently updated according to the newly-arrived requests).

An example of MEMSO route representation together with position and velocity coding is presented in Fig. 3. In MEMSO algorithm the velocity of each particle is calculated in real values, according to SPSO 2007 (see eq. (2)) and rounded to the nearest integer. This integer velocity vector is afterwards added to the integer state (position) vector modulo n, where



x_{t-1}	1	1	1	1	1	3	3	3	2	2
v_{t-1}	+0.2	+3.1	-0.4	-2.9	+1.2	+3.4	-2.8	-0.8	+2.8	-2.6
x_t	1	1	1	1	2	3	3	2	2	2

Fig. 3. An example solution for the problem with 10 requests and 3 vehicles. The depot is located in the center of the figure. The MEMSO particle position and velocity encoding are presented below the solution. The initial routes (with no longer active edges marked in dotted red) are $0-4-3-1-5-2-0$, $0-6-7-8-0$ and $0-10-9-0$, respectively. The routes after particles' movement (marked in black) are $0-4-3-1-2-0$, $0-6-7-0$ and $0-5-10-8-9-0$, respectively. Zeros in the routes represent the depot. In MEMSO encoding position is a $\mathbb{Z}_n^m$ vector and velocity is a $\mathbb{R}^m$ vector. Velocity is rounded to the nearest integer and added modulo n to the previous position.

n is the number of vehicles. This way a new position (vehicle assignment) is obtained.

Then the fitness function value is calculated as a sum of all vehicles routes' lengths optimized with the 2-OPT method:

$$COST_2(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{|r_i|} \rho(r_{i2OPT_j}, r_{i2OPT_j-1}) \qquad (12)$$

The MAPSO/MEMSO approach is the main point of reference for our algorithm and therefore its specificity and more detailed comparison with our approach will yet be discussed in several further sections devoted to presentation of particular aspects of these methods.

IV. Two-Phase Multi-Swarm PSO Method (2MPSO)

A very distinctive feature differing our approach from other CI-based methods is a clear separation of the clustering task (assignment of requests to vehicles) from the routes' optimization task (optimal ordering of requests assigned to a given vehicle). This section starts with a detailed presentation of the method, followed by its comparison with MEMSO/MAPSO and 2PSO (a single-swarm version published previously).

A. Pseudocode of 2MPSO

In each time step, the method considers the set C_t of the requests known at the time t and not ultimately assigned to

1: $\{C_t$ a set of requests known at time t which are not ultimately assigned$\}$
2: **while** $time \leq end$ **do**
3: $radius \Leftarrow 2 \max_{l_1,l_2=k+1,\ldots,k+m} \rho(l_1, l_2)$
4: **for all** $swarm \in swarms$ **do**
5: FIRST PHASE:
6: $\{heuristicSolution$ is created in order to increase particles' diversity and to keep solution in reasonable bounds$\}$
7: $heuristicSolution \Leftarrow CapacitatedMinimalSpanningForest(C_t)$
8: **if** $t = 0$ **then**
9: $bestSolution \Leftarrow heuristicSolution$
10: **end if**
11: {first particle in the $swarm$ is initialized with $heuristicSolution$ in order to keep $bestSolution$ in reasonable bounds}
12: {every tenth particle in the $swarm$ is initialized with the $bestSolution$ in order to preserve information about previous solution}
13: {all other particles in the $swarm$ are initialized within a given radius from the $bestSolution$}
14: $swarm <= InitializeSwarm(heuristicSolution, bestSolution, radius)$
15: **for** $i = 1, 2, \ldots, maxAssignmentIterations$ **do**
16: $PerformOptimizationStep(swarm)$
17: **end for**
18: $swarmBestSolution = GetBestSolution(swarm)$ $\{swarmBestSolution$ is treated as a set of vehicles with assigned initial routes$\}$
19: SECOND PHASE:
20: **for all** $vehicle \in swarmBestSolution$ **do**
21: $vehicle.route \Leftarrow EnhanceWith2OPT(vehicle.route)$
22: $vehicleSwarm \Leftarrow InitializeVehicleSwarm(vehicle.route)$
23: **for** $i = 1, 2, \ldots, maxRouteIteration$ **do**
24: $PerformOptimizationStep(vehicleSwarm)$
25: **end for**
26: $vehicle.route \Leftarrow GetBestRoute(vehicleSwarm)$
27: $vehicle.route \Leftarrow EnhanceWith2OPT(vehicle.route)$
28: **end for**
29: {reassign in a greedy way all requests violating time constraint of the problem (see eq. (9))}
30: $swarmBestSolution = RepairBestSolution(swarmBestSolution)$
31: {if the situation has not changed between t and $t-1$, we preserve the $bestSolution$ if it was better}
32: **if** $C_t \subseteq C_{t-1}$ **and** $swarmBestSolution > bestSolution$ **then**
33: $swarmBestSolution \Leftarrow bestSolution$
34: **end if**
35: **end for**
36: $bestSolution = ChooseBestSolution(swarms)$
37: **end while**

Fig. 4. High-level pseudocode of the 2MPSO algorithm.

any vehicle (although they may be a part of the proposed solution). Until the end of the day (line 2) a multi-swarm PSO optimization (line 4) is performed. For each $swarm$ a $heuristicSolution$ for assignment problem is constructed and part of the $swarm$ is initialized on the basis of this $heuristicSolution$ and based on the $bestSolution$ found in the previous step (or part of this solution as the number of necessary vehicles might be larger then in previous time step). Then for the $maxAssignmentIterations$ the system performs the PSO-based optimization (simultaneously for all $swarms$).

Subsequently, the $swarmBestSolution$ is chosen (line 18) and its routes are optimized (lines 20-28). First the 2-OPT algorithm is applied (which is an optional step if 2-OPT had already been used for assessing the lengths of possible routes in the assignment phase) and afterwards another PSO optimization is performed separately for each of the routes (lines 22-26). The best solution is additionally optimized with 2-OPT algorithm (line 27).

The final optimization procedure (line 30) applied to each swarm is aimed at repairing unfeasible routes (violating time constraint from eq. (9)) by means of greedy reassignment of requests. In addition the previous $bestSolution$ is used instead of a current $swarmBestSolution$ if the current set of clients is identical to previous one (lines 32-(line 33)).

At the end of a time step (in line 36) the new $bestSolution$ is chosen among all $swarmBestSolutions$. In the event scheduler module the $bestSolution$ is used to dispatch vehicles (which must be committed at this step due to time constraints). Furthermore, if more vehicles seem to be needed

in the next time step, the $bestSolution$ is extended by adding new vehicles and new (randomly positioned) vehicles' centers of operation. The number of necessary vehicles is estimated based on the $heuristicSolution$.

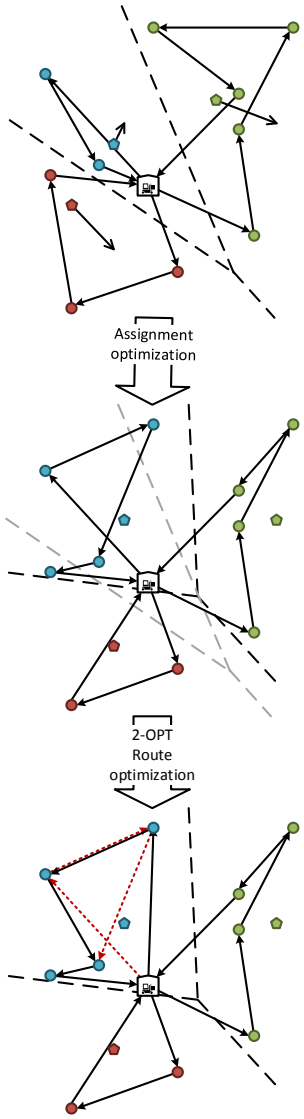


Fig. 5. Example of an assignment encoding and a position change of a single particle in one iteration of the first phase of the 2MPSO algorithm. The top figure presents the assignment space division (dashed lines) generated by a single particle having 6 dimensions corresponding to 3 cluster centers coordinates. Each cluster center is represented by a pentagon (there are three of them and each has a velocity vector denoted by an arrow). The middle figure presents the situation after updating the particles' position (gray dashed lines present the previous division/assignment of requests). Please note, that the insertion of new requests into the route is performed on the basis of requests rank in the previous assignment. The bottom figure presents the routes after the 2-OPT optimization. These routes (requests-to-vehicles assignments) will be evaluated by the PSO algorithm in the next iteration of the first phase. Dotted red arrows represent the initial route, before its enhancement by the 2-OPT method.

B. Auxiliary clustering heuristic

Please observe that in order to achieve higher quality of solutions in each time slice one of the PSO particles is initialized at the centers of clusters found by an auxiliary heuristic aimed at solving the clients-to-vehicles assignment problem (cf. lines 7-10 in Fig. 4). This heuristic uses the Kruskal algorithm [30] but, unlike in the original formulation, stops linking separate trees when the sum of weights of their nodes would exceed vehicle's capacity. Such an approach may be also interpreted as the modification of a procedure of hierarchical clustering with a single-linkage criterion[4].

All the remaining particles are initialized based on the $bestSolution$ found in the previous time slice or the Kruskal solution in the case of the first time slice (lines 12-13).

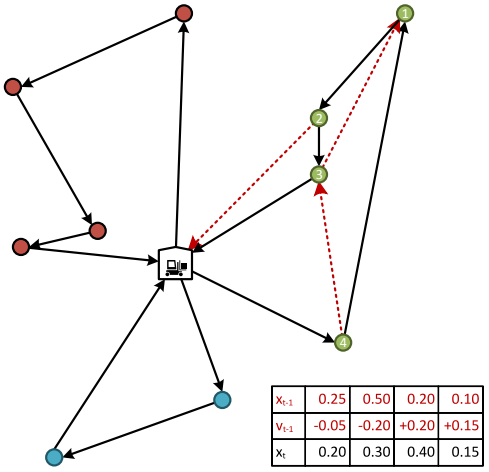


Fig. 6. Example of the route encoding in the second phase of the 2MPSO algorithm. One of the routes is composed of the assignments $(1, 2, 3, 4)$. Their order in the route at iteration $t-1$ (denoted in dotted red) was $0-4-3-1-2-0$, where 0 denotes the depot. This order is defined by the x_{t-1} values considered in the ascending order. Next, based on (2) the velocity v_{t-1} is calculated and added to the position vector, leading to a new position x_t representing the following order: $0-4-1-2-3-0$, which is induced by ordering the x_t elements.

C. First-Phase problem encoding

In the first phase each particle is represented as a real number vector whose elements denote centers of clusters of requests assigned to vehicles (see section IV-F). The area of clients' requests 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 assigned to each vehicle is a parameter of the 2MPSO algorithm.

The estimated solution assessment (fitness function) in the first phase is calculated as one of the following (see Fig. 7):

- a sum of distances from the inter-cluster requests to the clusters' centers (a measure of quality of a clustering) and the twice the distances from the clusters' centers to the depot location (a measure of a cost of reaching a cluster and returning to the depot in the end of a route).
- a total length of all vehicles' routes from the proposed clusters, formed with the use of 2-OPT.

The particle's position update procedure in the first phase is depicted in Fig. 5.

[4]Due to the space limits the Kruskal method is not revised here. Please consult [30] for its detailed formulation.

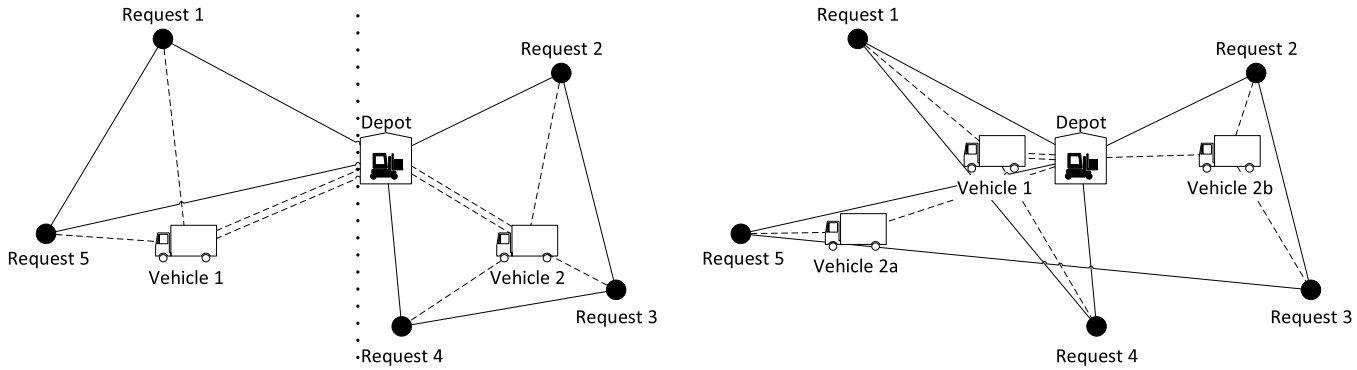


Fig. 7. Example of a DVRP problem with 2 vehicles and 5 clients' requests. Solid lines represent possible routes, whose lengths are used as an evaluation function by MEMSO, $2MPSO_2$ and $2MPSO_3$. Dashed lines represent estimated cluster cost (quality), which is used as an evaluation function by $2MPSO_1$. Dotted line separates the two operating areas assigned to vehicles. Right figure shows a solution which is non-representable by single cluster per vehicle encoding used by $2MPSO_1$ and $2MPSO_2$ since the sets of requests assigned to vehicles are not linearly-separable.

In the first case the cost function is of the form (13):

$$COST_1(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=1}^{|r_i|} \rho(r_{ij}, i) \qquad (13)$$

where i represents the coordinates of the estimated center of the cluster of requests assigned to the ith vehicle. In the latter case the $COST_2$ function (the same one as in MEMSO) is applied.

D. Second-Phase problem encoding

In the second phase each particle represents the order of requests assigned to a given vehicle (each cluster/vehicle is solved by a separate PSO instance). The order is obtained by sorting coordinates of each of the particles in the ascending order (cf. Fig. 6).

The solution assessment in the second phase (in each of the PSO instances) 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 PSO instances.

E. Three versions of the 2MPSO method

In the experiments reported in this paper the following three combinations of the fitness functions and first-phase encodings of the DVRP were considered:

- $2MPSO_1$ - the algorithm with $COST_1$ evaluation function and one cluster of requests per vehicle.
- $2MPSO_2$ - the algorithm with $COST_2$ evaluation function and one cluster of requests per vehicle.
- $2MPSO_3$ - the algorithm with $COST_2$ evaluation function and up to three clusters of requests per vehicle.

Let us denote by m the number of requests ($50 \leq m \leq 199$ in the benchmarks used), by n the number of available vehicles (n was equal to 50 in all tested benchmarks), by $\hat{n}$ the estimated number of required vehicles (usually around $\frac{n}{3}$), by c the number of clusters per vehicle ($c = 1$ was used in $2MPSO_1$ and $2MPSO_2$, and $c = 3$ in $2MPSO_3$). Additionally, let us denote by b the number of spare vehicles (with respect to the estimation provided by the single-linkage clustering algorithm - cf. section IV-B) - we used $b = 4$ based on results of the pilot study. The theoretical and the experimental dimension sizes of the three variants of 2MPSO method and the MEMSO method are as follows:

Algorithm	Type	Theoret. size	Exper. size
MEMSO	Discrete	m	$[50, 199]$
$2MPSO_1$	Continuous	$2n$	100
$2MPSO_2$	Continuous	$2(\hat{n} + b)$	$[20, 40]$
$2MPSO_3$	Continuous	$2 \cdot 3(\hat{n} + b)$	$[60, 120]$

F. Comparison of MEMSO and 2MPSO Encodings

For a better explanation of the differences between MEMSO and 2MPSO requests assignment encodings, Fig. 7 presents an example of DVRP with 2 vehicles and 5 requests. The encodings of MEMSO and 2MPSO are as follows:
For the left subfigure:

$MEMSO$	1 | 2 | 2 | 2 | 1
$2MPSO_{1/2}$	Veh1$_x$ | Veh1$_y$ || Veh2$_x$ | Veh2$_y$

For the right subfigure:

$MEMSO$	1 | 2 | 2 | 1 | 2
$2MPSO_{1/2}$	Encoding does not exist (assignments are not linearly-separable)

where Veh$n_k, n = 1, 2, k = x, y$ denotes coordinate k of the nth cluster center.

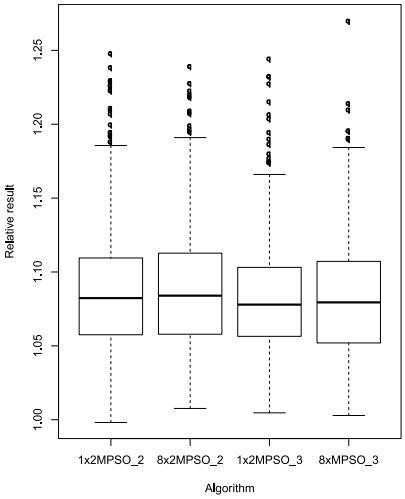
In addition, Fig. 7 also presents the way the fitness function is calculated in each of the algorithms. In MEMSO, $2MPSO_2$ and $2MPSO_3$ algorithms it is the sum of solid-line edges while in the $2MPSO_1$ algorithm the sum of dashed-line edges.

G. Multi-swarm approach vs. single-swarm one

Our initial approach to solving DVRP, the 2PSO method proposed in [15] was a single-swarm PSO implementation

TABLE I
COMPARISON OF MULTI-SWARM ALGORITHMS ($2MPSO_2, 2MPSO_3$) WITH A SINGLE-SWARM APPROACH ($2PSO_2, 2PSO_3$) WITH ADEQUATELY ENLARGED SWARM SIZE. FOR EACH BENCHMARK THE BEST RESULT AMONG THE TESTED METHODS IS BOLDED. THE BOXPLOTS PRESENT THE SUMMARY OF THE LENGTH OF SOLUTIONS FOR SINGLE AND MULTI-SWARM ALGORITHMS NORMALIZED BY THE BEST KNOWN RESULT FOR EACH OF THE BENCHMARKS.

	$2PSO_2$ $(50*1*(4*10^4))$		$2MPSO_2$ $(50*8*(0.25*10^4))$		$2PSO_3$ $(50*1*(4*10^4))$		$2MPSO_3$ $(50*8*(0.25*10^4))$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	571.35	593.97	**558.07**	600.54	566.14	595.50	569.23	**591.55**
c75	**877.30**	921.90	894.96	926.52	884.99	**913.07**	888.66	920.68
c100	904.52	959.15	918.79	961.61	919.64	960.73	**898.16**	**958.90**
c100b	826.47	851.70	827.85	860.38	**825.14**	**847.53**	825.65	859.64
c120	1068.73	1129.74	1068.69	1110.71	**1059.31**	**1103.18**	1072.54	1123.45
c150	**1114.90**	1170.14	1129.86	1185.54	1125.64	**1167.95**	1123.10	1170.00
c199	1384.92	1447.94	1389.48	1450.49	1406.54	1443.75	**1383.07**	**1429.01**
f71	279.83	**295.32**	280.78	298.32	**279.64**	299.32	279.95	298.86
f134	12208.09	12860.23	11997.78	12541.47	12216.92	12918.91	**11984.50**	**12525.63**
tai75a	1724.58	1858.52	1739.35	1872.40	**1716.76**	**1847.95**	1744.01	1873.52
tai75b	1451.89	1536.39	1455.66	1530.52	1450.98	1524.83	**1428.16**	**1521.33**
tai75c	1465.90	1568.25	**1465.14**	1569.06	1481.41	1551.81	1470.97	**1550.25**
tai75d	1432.47	1479.90	1434.41	1474.14	1430.21	**1462.88**	**1422.94**	1471.94
tai100a	2198.02	2310.03	**2195.04**	**2298.33**	2209.33	2329.67	2222.80	2303.29
tai100b	2095.52	2172.50	**2064.50**	**2171.76**	2084.03	2179.42	2068.86	2176.57
tai100c	1527.46	1583.03	1517.45	1596.35	**1509.79**	**1579.96**	1526.07	1588.78
tai100d	1712.53	1793.97	1746.34	1821.29	**1705.47**	**1783.77**	1742.76	1813.23
tai150a	3529.25	3751.09	3495.08	3731.86	3438.45	3677.48	**3395.02**	**3658.90**
tai150b	**2988.54**	3132.40	3072.90	3152.98	3017.96	3148.42	3017.35	**3119.40**
tai150c	**2620.78**	2789.79	2624.62	2777.28	2631.29	2790.87	2631.67	**2754.52**
tai150d	**2942.62**	3141.15	2996.00	3135.51	2963.41	**3106.85**	3023.47	3122.89



relying on one swarm composed of 40 particles, whereas the 2MPSO is a multi-swarm approach with 8 swarms consisting of 40 particles each.

The obvious advantage of a multi-swarm design is using 8 times greater number of particles, but what could be even more important, using several largely independent swarms may potentially allow better exploration of the search space. In order to verify whether the use of multi-swarm approach actually improves the algorithm's performance, we have tested the $2MPSO_2$ and $2MPSO_3$ with 8 swarms (each composed of 40 particles) against their $2PSO$ counterparts (with 1 and 3 clusters per vehicle, respectively) with one swarm composed of 320 particles.

In order to assure the same level of overall internal particles' connectivity in all tested variants the following reasoning was applied. Since in a multi-swarm approach a probability that particle x is a neighbor of particle y equals 0.5, in the single (large) swarm approach, for each particle a set of 40 potential neighbors was randomly selected and then the actual neighbors were chosen with probability 0.5 from this selection. No neighbors were defined outside this set.

The comparison results are presented in Table I. In short, out of 21 benchmarks the $2PSO_2$, $2MPSO_2$, $2PSO_3$ and $2MPSO_3$ were the winners (in terms of finding the shortest solution) in 5, 4, 6 and 6 problems respectively. In terms of the best average (based on 30 trials per test set) the numbers of wins were equal to 1, 2, 9 and 9, respectively. Since no significant differences in average results between the single-swarm and the multi-swarm methods were observed, the latter one remained our approach of choice due to more efficient parallelization, allowing for nearly 8 times speed gain. Clearly, in both cases the methods with up to 3 clusters per vehicle are superior over their 1 cluster counterparts.

Another difference between a single- and multi- swarm versions is a need for problems instances synchronization between swarms in the latter case. This issue is discussed below in section IV-I.

H. Novelty of the 2MPSO method

In our initial Two-Phase Particle Swarm Optimization (2PSO) approach [15], we used standard (continuous) version of the single-swarm PSO and proposed splitting the process of solving DVRP into two phases: a clustering phase, in which requests are assigned to particular vehicles, and an ordering phase, in which the tour for each vehicle was found with the use of (a separate instance of) the PSO algorithm. In 2MPSO approach we extend that method by applying a multi-swarm approach and testing, in the clustering phase, different fitness functions and different DVRP encodings.

Moreover, for the first time in the literature a single-linkage capacitated clustering method / minimum spanning forest is used (line [6] in Fig. 4) instead of a greedy or random assignment, as an auxiliary heuristic in swarm initialization procedure at each time slice.

A novelty of proposed approach also lies in the possibility of assigning more than one cluster of requests per vehicle. In effect, the algorithm takes into account a trade-off between a simple one-cluster heuristic (which assigns pairwise close requests to the same vehicle) and multi-cluster approach which allows more flexible assignment, however, at the cost of increasing a search space dimension (the size of a vector representing the cluster centers).

Our method also differs significantly from the works of Khouadjia et al. [4], [26], [17] which use a discrete coding of the DVRP, hence a version of the PSO suitable for discrete search spaces.

Additionally, our method is methodologically simpler than MEMSO as we avoid the necessity of direct on-line swarm synchronization. Unlike in the comparative algorithm, where some number of particles was migrated between swarms, we

maintain the swarms separately with no communication within a single slice time span.

In our opinion, the main incentive of using the 2MPSO method is its generality, which stems from its "natural capacity" to incorporate other than PSO continuous optimization metaheuristics in both phases. Thanks to the proposed problem encoding and the lack of specific memory requirements (only the hitherto best solution is maintained between slices) the use of the PSO as the baseline method is not mandatory.

Finally, to the best of our knowledge, the two-phase approach was previously applied in the vehicle routing optimization literature exclusively for solving the static version (VRP) of the problem [6], [5], [7].

I. Knowledge Transfer and Swarms Synchronization

In dynamic problems, one of the crucial tasks is efficient transfer of knowledge from partial (incomplete) problems to the final solution. Generally, it is assumed that solutions obtained for the two problem instances which are close in time should not differ much and therefore knowledge transfer may, in principle, be very advantageous.

Another critical issue is the efficient usage of parallel or distributed architecture and knowledge transfer between partial problems within the same problem instance (time slot). Both discussed PSO-based methods (i.e. MAPSO/MEMSO and 2MPSO) deal with these two issues differently.

Knowledge Transfer Between Time Slices: In the MEMSO algorithm it is proposed to add a memory to each particle in order to store its best known solution (the vector of vehicles' identifiers assigned to requests) from the previous time slice. When new requests arrive, they are processed in a random order and assigned to vehicles by a greedy algorithm, thus forming the initial swarm locations for the PSO method.

In the 2MPSO algorithm a different approach is taken. Since the solution of the first phase in the previous time slot consists of locations of clusters centers, these coordinates are treated as reliable estimations of the clusters centers after the arrival of new requests (in the next time slice). Therefore initial swarm location is defined around the center of the previous best known solution within a given radius.

Please note that the above-described single-solution-based knowledge transfer is, in principle, more general than that of the MAPSO/MEMSO as it allows for using virtually any optimization algorithm for finding the currently best solution, not necessarily the population-based method.

Knowledge Transfer Between Swarms/Swarms Synchronization: In the MEMSO algorithm knowledge is transferred between swarms by migrating particles. In every iteration each particle can migrate with some small probability to another swarm. As MEMSO allows for a distributed way of solving the problem there is, in general, no guarantee that at a given moment all swarms are solving the same problem instance (from the same time slice). Therefore a particle after migration may need to wait to be incorporated into a new swarm or must be re-initialized with newly received requests.

The 2MPSO algorithm assumes that the problem is solved in a parallel way on a single multi-threaded computer. Therefore, we take an easy approach in which, within a given time slice, each thread works in isolation and at the end of allotted time (slice time span) all threads are synchronized and the best solution found is spread out among the threads. Such an approach is motivated by an assumption that at the end of a time slice some vehicles are already committed to serve certain sets of requests and it would be meaningless to solve problem instances not synchronized with the current state of the problem instance.

V. Experimental setup and benchmark problems

A. Benchmark files

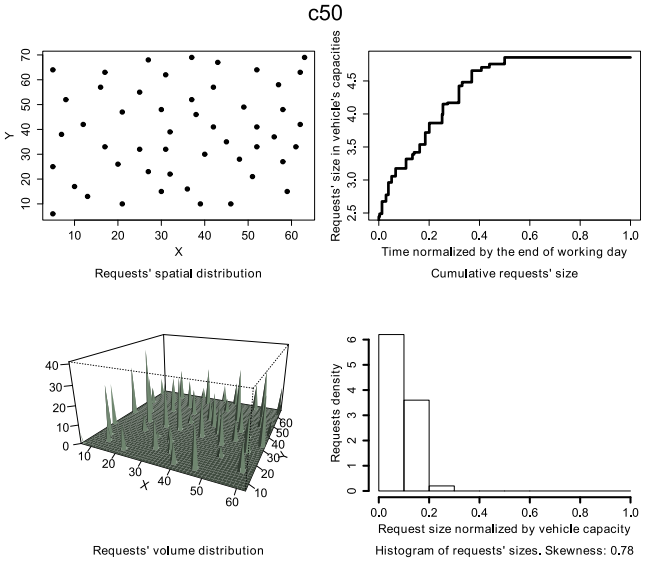


Fig. 8. Spatial and volume distribution of requests (left subfigures), the plot of cumulative requests' size and the histogram of requests' sizes (right subfigures) of the **c50** benchmark set. **c50** is characterized by a uniform spatial distribution of requests that are relatively small and similar in size.

In order to evaluate the performance of the algorithm we used dynamic versions of Christofides' [31], Fisher's [32] and Taillard's [7] benchmark sets [33]. Each instance consists of between 50 and 199 requests to be served by a fleet of 50 vehicles (the number of requests is included in 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 very well clustered problems, semi-clustered ones, and completely unstructured instances. Also the volume distribution and skewness significantly differ between benchmarks.

Spatial and volume distributions as well as histogram of requests' sizes and the plot of their cumulative size over time in the three exemplar benchmark sets (one from each of the three sources) are presented in Figs. 8, 9 and 10.

B. Algorithm's parameters

The following selections of parameters were tested in the experiments (default values are bolded):

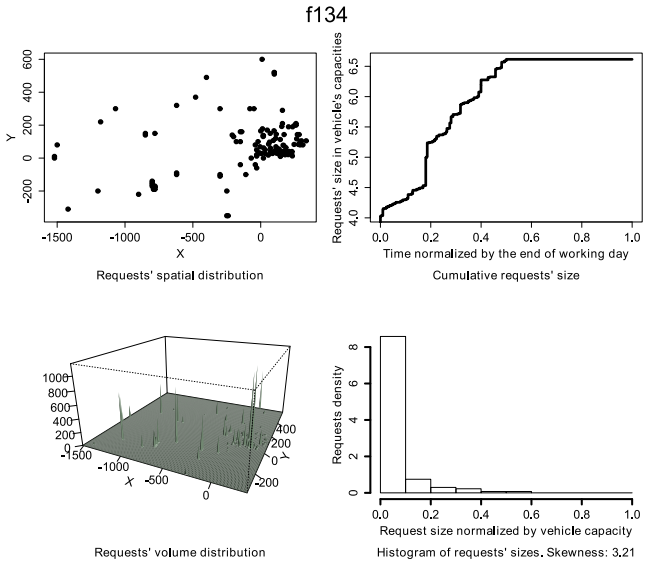


Fig. 9. Spatial and volume distribution of requests (left subfigures), the plot of cumulative requests' size and the histogram of requests' sizes (right subfigures) of the **f134** benchmark set. **f134** is characterized by a semi-structured spatial distribution of requests with quite a high number of relatively small requests and several large requests appearing within the first 20% of a working day time.

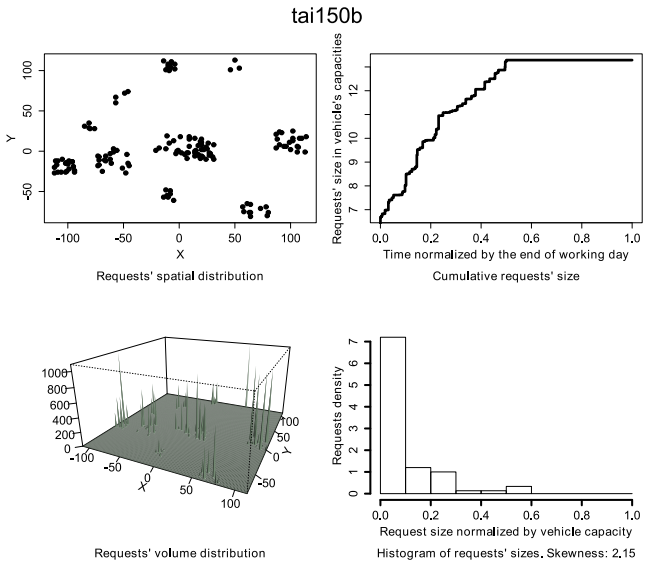


Fig. 10. Spatial and volume distribution of requests (left subfigures), the plot of cumulative requests' size and the histogram of requests' sizes (right subfigures) of the **tai150b** benchmark set. **tai150b** is characterized by a clustered spatial distribution of requests with a relatively high number 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.

Parameter	Value(s)
g	0.60
l	2.20
a	0.63
$P(X$ is a neighbor of $Y)$	0.50
#clusters	$\{1, \mathbf{3}\}$
#iterations 1st/2nd phase	$\{10/2, \mathbf{50/12}, 200/50\}$
#particles	$\{20, \mathbf{40}, 100, 320\}$
#swarms	$\{1, \mathbf{8}\}$
T_{co}	0.5
n_{ts}	$\{25, \mathbf{50}, 100, 320\}$
T_{ac}	$2 * \frac{end - start}{n_{ts}}$

The values of g , l, a and P where chosen experimentally based on some number of initial tests. The rationale behind the remaining parameters' selections was discussed in the previous sections.

VI. Results

The properties and the efficacy of the proposed 2MPSO method were experimentally verified in the three main experiments. First of all, the three variants of 2MPSO were compared with the competitive CI-based metaheuristic approaches discussed in section III for approximately the same number of FFE. In the next experiment the potential for further improvement of 2MPSO results in the case of increased number of FFE per time slice was tested. Finally, the third set of tests aimed at assessing the relevance of particular 2MPSO components and their contribution to the overall efficacy of the method.

A. Comparison of 2MPSO with MEMSO/MAPSO and other CI-based approaches

In the main experiment all methods introduced in section III were compared with 2MPSO for the same number of FFE. Special emphasis was put on a direct comparison between three versions of the 2MPSO algorithm and the MAPSO/MEMSO approach, which provided majority of the best individual and the best average literature results so far. The number of swarms in all these multi-swarm methods was equal to 8.

The results are presented in Table II. The best individual and the best average results are bolded[5]. It can be seen from Table II that 2MPSO with $COST_2$ evaluation function and 3 clusters per vehicle (denoted $2MPSO_3$) outperforms $MEMSO$ by approximately 1.5% in the average result and is able to find new best solutions in 7 out of 21 benchmark instances (30 experiments per benchmark were performed). The same method with 1 cluster per vehicle ($2MPSO_2$) gained a 1% advantage in terms of the average results and was able to improve the best literature solutions for 3 tested instances. $2MPSO_1$ found the new best solution for one test instance ($c120$).

Generally speaking, $2MPSO_3$ and $MEMSO$ appeared to be the best two methods in the presented evaluation. The methods found new best individual results for $7/5$ test instances, respectively and each of them attained 6 best average results. Furthermore, $2MPSO_3$ also attained the lowest sum-of-minimum and sum-of-averages values among all 21 benchmarks. In a direct comparison $2MPSO_3$ accomplished statistically significantly better results than $MEMSO$ in 7 cases, while in the 8 benchmarks the opposite situation took place (cf. the shaded results in Table II).

[5]For the sake of space savings and clarity of the presentation the results for the ACS method [14] are not presented as ACS appeared to be inferior to all other methods and in none of the benchmarks was found to be the most efficient approach.

TABLE II

COMPARISON BASED ON 10^6 FITNESS FUNCTION EVALUATIONS. THE NUMBERS IN THE PARENTHESES RESPECTIVELY DENOTE: THE NUMBER OF TIME SLICES, THE NUMBER OF SWARMS AND THE NUMBER OF FFE WITHIN EACH SWARM IN EACH TIME SLICE. IN THE CASE OF THE $2MPSO_1$ METHOD THE NUMBER OF FFES IS SLIGHTLY INCREASED COMPARED TO THE COMPETITIVE ALGORITHMS SO AS TO MAKE A TIME-COMPENSATION FOR THE LACK OF THE 2-OPT OPTIMIZATION ROUTINE IN THIS METHOD. THE LAST TWO APPROACHES (GA AND TS) DID NOT USE THE NUMBER OF FFES BUT THE RUNNING TIME AS EFFICIENCY METRICS [13]. IN THE TABLE, GRAY BACKGROUND DENOTES RESULTS STATISTICALLY SIGNIFICANTLY DIFFERENT IN $2MPSO_3$ VS. $MEMSO$, GIVEN BY WILCOXON SIGNED RANK TEST.

	$2MPSO_1$ $(50*8*(0.3*10^4))$		$2MPSO_2$ $(50*8*(0.25*10^4))$		$2MPSO_3$ $(50*8*(0.25*10^4))$		$MAPSO$ $(25*8*(0.5*10^4))$		$MEMSO$ $(25*8*(0.5*10^4))$		GA $(25*30$ seconds$)$		TS $(25*30$ seconds$)$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	591.82	638.04	**558.07**	600.54	569.23	**591.55**	571.34	610.67	577.60	592.95	570.89	593.42	603.57	627.90
c75	903.38	950.76	894.96	926.52	**888.66**	**920.68**	931.59	965.53	928.53	962.54	981.57	1013.45	981.51	1013.82
c100	977.57	1058.68	918.79	961.61	**898.16**	**958.90**	953.79	973.01	949.83	968.92	961.10	987.59	997.15	1047.60
c100b	828.06	**839.59**	827.85	860.38	**825.65**	859.64	866.42	882.39	864.19	878.81	881.92	900.94	891.42	932.14
c120	**1068.41**	**1097.20**	1068.69	1110.71	1072.54	1123.45	1223.49	1295.79	1164.63	1284.62	1303.59	1390.58	1331.80	1468.12
c150	1165.50	1231.93	1129.86	1185.54	**1123.10**	**1170.00**	1300.43	1357.71	1274.33	1327.24	1348.88	1386.93	1318.22	1401.06
c199	1399.20	1455.68	1389.48	1450.49	**1383.07**	**1429.01**	1595.97	1646.37	1600.57	1649.17	1654.51	1758.51	1750.09	1783.43
f71	311.84	358.43	280.78	298.32	**279.95**	298.86	287.51	296.76	283.43	**294.85**	301.79	309.94	280.23	306.33
f134	12146.30	**12486.97**	11997.78	12541.47	**11984.50**	12525.63	15150.50	16193.00	14814.10	16083.82	15528.81	15986.84	15717.90	16582.04
tai75a	1812.73	1933.26	**1739.35**	1872.40	1744.01	1873.52	1794.38	1849.37	1785.11	**1837.00**	1782.91	1856.66	1778.52	1883.47
tai75b	1503.60	1662.52	1455.66	1530.52	1428.16	1521.33	**1396.42**	1426.67	1398.68	**1425.80**	1464.56	1527.77	1461.37	1587.72
tai75c	1513.00	1651.96	1465.14	1569.06	1470.97	1550.25	1483.10	1518.65	1490.32	1532.45	1440.54	**1501.91**	**1406.27**	1527.80
tai75d	1465.89	1538.89	1434.41	1474.14	1422.94	1471.94	1391.99	**1413.83**	**1342.26**	1448.19	1399.83	1422.27	1430.83	1453.50
tai100a	2247.52	2394.65	2195.04	2298.33	2222.80	2303.29	2178.86	2214.61	**2170.54**	**2213.75**	2232.71	2295.61	2208.85	2310.37
tai100b	2117.12	2299.38	**2064.50**	**2171.76**	2068.86	2176.57	2140.57	2218.58	2093.54	2190.01	2147.70	2215.39	2219.28	2330.52
tai100c	1609.72	1718.61	1517.45	1596.35	1526.07	1588.78	**1490.40**	**1550.63**	1491.13	1553.55	1541.28	1622.66	1515.10	1604.18
tai100d	1864.00	1977.35	1746.34	1821.29	1742.76	**1813.23**	1838.75	1928.69	**1732.38**	1895.42	1834.60	1912.43	1881.91	2026.76
tai150a	3536.38	3834.82	3495.08	3731.86	3395.02	3658.90	3273.24	3389.97	**3253.77**	**3369.48**	3328.85	3501.83	3488.02	3598.69
tai150b	3070.10	3208.90	3072.90	3152.98	3017.35	3119.40	**2861.91**	**2956.84**	2865.17	2959.15	2933.40	3115.39	3109.23	3215.32
tai150c	2738.88	2840.73	2624.62	2777.28	2631.67	2754.52	2512.01	2671.35	**2510.13**	**2644.69**	2612.68	2743.55	2666.28	2913.67
tai150d	3049.06	3265.03	2996.00	3135.51	3023.47	3122.89	**2861.46**	**2989.24**	2872.80	3006.88	2950.61	3045.16	2950.83	3111.43

B. Possible room for 2MPSO results improvement

The next experiment aimed at verification, whether using higher numbers of FFE per time slice, in the case of the $2MPSO_3$ method (the most effective variant of $2MPSO$) would lead to further improvement.

The results presented in Table III, show that increasing the number of FFE by a factor of 10 per time slice (the third double column of the table) was truly beneficial and allowed for improving the average results for all but two benchmarks compared to the previous number of FFE (the second double column of the table). Further 10-times increase of the number of FFE resulted in yet better performance and led to the average results improvement in 17 test sets and allowed finding new best literature results for 8 benchmarks (cf. Table V).

C. Contributions of particular methods

One of the critical research questions is the estimation of contributions of particular components of the proposed 2MPSO method. The significance of multi- versus single swarm approach has already been reported in section IV-G. In this section we discuss the performance of the method with particular components being switched off. Experimental results are presented in Table IV, where $2MPSO_3$ denotes the "complete" implementation of the 2MPSO method with 3 clusters per vehicle, $2MPSO_3 - Tree$ does not use auxiliary swarms initialization provided by the modified Kruskal algorithm, $2MPSO_3 - 2ndPhase$ does not use the second phase of the method (i.e. the routes are built directly on clusters found in the first phase by means of the 2-OPT method), $Tree + 2OPT$ represents the method (not using PSO at all) in which clusters are build based on the modified Kruskal algorithm and the routes are constructed with 2-OPT, afterwards.

It can be seen from Table IV that leaving out any of the $2MPSO$ components generally causes deterioration of results. While a complete $2MPSO_3$ method accomplished to find 11 best minima and 10 best average scores among 21 tested benchmarks, the least harmful appeared an exclusion of the 2nd phase of the algorithm (7 minima and 7 best averages) followed by omitting the additional initialization of one particle (solution) by the modified Kruskal algorithm (3 minima and 3 best averages). The exclusive use of both supportive methods (Kruskal algorithm and 2-OPT) without the main PSO-based engine led to achieving a single best average value.

VII. CONCLUSIONS AND FUTURE WORK

The paper describes and experimentally evaluates the PSO-based algorithm for solving Dynamic Vehicle Routing Problem. In the considered version of DVRP clients' requests may arrive during the whole working day, which makes solving the problem a challenging task.

In our method, similarly to other approaches reported in the literature, the whole day period is divided into a certain number of time slices and in each of them a static, currently available instance of the Vehicle Routing Problem is solved. However, unlike in majority of the CI-based approaches reported in the literature, in each time slice we split the solving process into two separate phases: assignment of requests to vehicles (clustering phase) and route construction for each of the vehicles (optimization phase). Each of them is solved by a separate PSO instance. The overall solution implements simple, but effective, mechanisms of knowledge transfer between multiple swarms (in-phase), as well as, between consecutive phases.

Experimental evaluation on a well established set of benchmark problems shows potential of our method which, for

TABLE III

COMPARISON OF MULTI-SWARM ALGORITHMS WITH THREE CLUSTERS OF REQUESTS PER VEHICLE AND VARIOUS NUMBER OF FFE PER TIME SLICE. THE BEST RESULTS AMONG PRESENTED ALGORITHMS ARE BOLDED. THE NEW BEST LITERATURE RESULTS ARE ADDITIONALLY MARKED IN GRAY. THE BOXPLOTS PRESENT THE SUMMARY OF THE LENGTH OF SOLUTIONS FOR PRESENTED NUMBER OF FFES NORMALIZED BY THE BEST KNOWN RESULT FOR EACH OF THE BENCHMARKS.

	$2MPSO_3$ $(50*8*(0.25*10^3))$		$2MPSO_3$ $(50*8*(0.25*10^4))$		$2MPSO_3$ $(50*8*(0.25*10^5))$		$2MPSO_3$ $(50*8*(0.25*10^6))$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	569.44	610.19	569.23	591.55	**552.47**	592.97	558.41	**591.29**
c75	927.58	983.32	888.66	920.68	880.62	905.62	**878.93**	**904.12**
c100	935.69	1013.22	898.16	958.90	891.01	932.35	**874.20**	**920.84**
c100b	828.26	880.45	825.65	859.64	822.81	847.22	**819.56**	**839.01**
c120	1071.08	**1114.08**	1072.54	1123.45	1058.03	1144.31	**1056.28**	1131.30
c150	1181.58	1283.11	1123.10	1170.00	**1096.53**	1137.57	1097.01	**1131.78**
c199	1469.70	1540.48	1383.07	1429.01	1366.79	**1403.31**	**1362.84**	1415.58
f71	**278.65**	308.13	279.95	298.86	280.17	297.41	282.69	**291.61**
f134	12586.83	12925.30	11984.50	12525.63	11970.05	12311.88	**11755.58**	**12038.28**
tai75a	1783.83	1980.62	1744.01	1873.52	1704.59	1785.67	**1682.90**	**1766.68**
tai75b	1523.66	1653.11	1428.16	1521.33	1416.25	1463.35	**1404.29**	**1447.30**
tai75c	1568.29	1663.28	1470.97	1550.25	1470.59	1519.58	**1406.59**	**1498.18**
tai75d	1454.32	1563.18	1422.94	1471.94	**1406.77**	1441.55	1415.79	**1439.78**
tai100a	2367.61	2525.70	2222.80	2303.29	2166.78	2233.04	**2155.42**	**2215.44**
tai100b	2199.71	2353.51	2068.86	2176.57	**2022.13**	**2079.61**	2034.82	2102.15
tai100c	1591.10	1686.45	1526.07	1588.78	1470.67	1516.87	**1446.10**	**1493.59**
tai100d	1863.38	1964.80	1742.76	1813.23	**1688.87**	**1747.21**	1690.32	1753.17
tai150a	3668.87	4012.98	3395.02	3658.90	3355.29	3471.60	**3298.91**	**3394.92**
tai150b	3087.92	3321.01	3017.35	3119.40	2928.43	3013.39	**2887.88**	**2992.81**
tai150c	2723.78	2930.53	2631.67	2754.52	2529.56	2615.09	**2462.96**	**2551.58**
tai150d	3107.45	3331.85	3023.47	3122.89	2913.91	3003.85	**2886.12**	**2941.66**

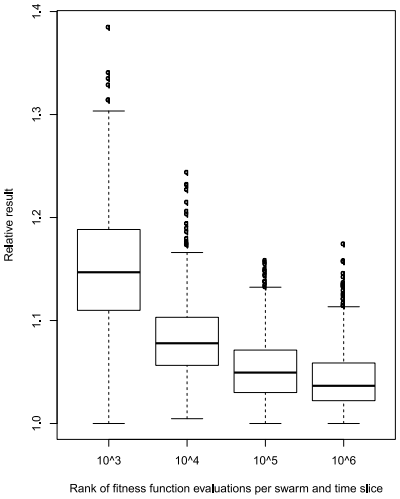
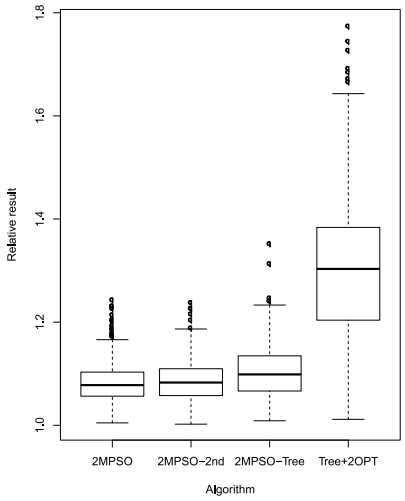


TABLE IV

COMPARISON OF MULTI-SWARM ALGORITHMS WITH THREE CLUSTERS OF REQUESTS PER VEHICLE AND AUXILIARY HEURISTICS OR PSO SWITCHED OFF. THE NUMBER OF FFES IN THE CASE OF $2MPSO_3 - 2nd\ Phase$ IS REDUCED PROPORTIONALLY TO THE LACK OF THE 2ND PHASE OF THE $2MPSO_3$ ALGORITHM, WHICH TAKES $12/62$ (APPROXIMATELY 20%) OF THE WHOLE FFE BUDGET. THE BOXPLOTS PRESENT THE SUMMARY OF THE LENGTH OF SOLUTIONS FOR PRESENTED MODULES OF THE ALGORITHM SWITCHED OFF NORMALIZED BY THE BEST KNOWN RESULT FOR EACH OF THE BENCHMARKS.

	$2MPSO_3$ $(50*8*(0.25*10^4))$		$2MPSO_3-2ndPhase$ $(50*8*(0.2*10^4))$		$2MPSO_3-Tree$ $(50*8*(0.25*10^4))$		$Tree+2OPT$ $(50*8)$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	569.23	**591.55**	**562.52**	592.10	563.43	620.97	683.82	788.68
c75	**888.66**	**920.68**	903.71	931.04	902.55	949.10	1101.34	1196.15
c100	**898.16**	958.90	915.83	**957.52**	932.31	1009.22	1139.28	1267.02
c100b	825.65	859.64	**821.34**	861.62	828.21	838.46	828.98	**830.18**
c120	1072.54	1123.45	1067.06	1112.20	**1065.55**	**1091.58**	1087.50	1121.09
c150	**1123.10**	**1170.00**	1126.75	1178.81	1130.11	1216.68	1321.63	1496.52
c199	1383.07	**1429.01**	**1377.31**	1429.67	1389.69	1478.81	1681.98	1842.93
f71	**279.95**	298.86	281.05	**298.07**	311.06	328.98	348.76	411.13
f134	11984.50	12525.63	**11821.38**	12432.38	12007.06	**12225.46**	13650.44	14219.83
tai75a	**1744.01**	1873.52	1777.82	**1872.83**	1753.90	1877.36	1971.69	2142.12
tai75b	**1428.16**	**1521.33**	1476.88	1532.70	1459.49	1539.87	1623.01	1690.74
tai75c	**1470.97**	**1550.25**	1477.22	1568.69	1483.88	1599.72	1832.35	2009.34
tai75d	**1422.94**	1471.94	1430.03	**1467.10**	1436.70	1478.04	1431.89	1624.71
tai100a	2222.80	**2303.29**	**2213.08**	2325.78	2213.77	2329.81	2501.11	2916.95
tai100b	2068.86	**2176.57**	2096.12	2180.66	**2047.24**	2274.96	2337.91	2656.51
tai100c	**1526.07**	1588.78	1544.44	1602.49	1531.29	**1584.37**	1780.80	2000.34
tai100d	1742.76	1813.23	**1728.51**	**1813.22**	1750.73	1836.15	2166.62	2366.10
tai150a	**3395.02**	**3658.90**	3492.33	3712.50	3427.56	3702.06	3802.01	4069.03
tai150b	**3017.35**	3119.40	3033.92	**3113.32**	3059.03	3140.54	3308.89	3657.68
tai150c	2631.67	2754.52	**2585.53**	**2742.37**	2621.97	2798.93	2774.54	3003.10
tai150d	3023.47	**3122.89**	3023.67	3161.49	**3020.70**	3127.68	3265.74	3559.72



comparable number of FFE, was capable of finding new best solutions in 11 out of 21 test instances. Further increase of the number of FFE led to finding yet few more best literature results, with the final outcome being equal to 17. Table V summarizes all best results reported for considered benchmarks together with the respective methods and their parameters. Graphical representation of the routes and vehicles' timetables corresponding to $2MPSO_{2/3}$ best results are available at our project's website [34].

Besides promising numerical results this work provides several qualitative conclusions related to higher-level observations of the effectiveness of the PSO method in the discussed DVRP embodiment. First of all, our studies indicate that the number of time slices initially proposed by Kilby et al. (25) is not always an optimal choice and should be adjusted for the selected optimization algorithm. In many cases increasing this figure to 50 (with accompanying respective decrease in the number of internal in-slice calculations of the FFE) significantly improves the results.

Secondly, as came out from the tests, different test problems can visibly benefit from the appropriate choice of the evaluation function and problem encoding used for temporal evaluation of the partial solutions assessed by the method. Three such approaches were proposed and verified in this paper with apparent impact on the quality of obtained solutions depending on the specificity of a particular test instance.

Thirdly, as came out from the tests, using the multi-swarm approach is, in general, more beneficial than application of a single swarm with respectively higher number of particles. The higher efficiency of several smaller swarms is mainly at-

TABLE V
BEST OVERALL RESULTS ACHIEVED FOR THE BENCHMARK SETS. THE RIGHTMOST COLUMN PRESENTS THE PARTICULAR ALGORITHM AND THE NUMBER OF THE EVALUATION FUNCTION CALCULATIONS DURING THE WHOLE EXECUTION GIVEN IN THE FORM SWARMS × TIME SLICES × EVALUATIONS PER TIME SLICE. THE NEW BEST SOLUTIONS COMPUTED BY $2MPSO$ CAN BE FOUND IN [34].

Name	Best result	Algorithm
c50	552.47	$2MPSO_3\ 8*50*0.25*10^5$
c75	877.30	$2MPSO_2\ 1*50*0.4*10^5$
c100	874.20	$2MPSO_3\ 8*50*0.25*10^6$
c100b	819.56	$2MPSO_3\ 8*50*0.25*10^6$
c120	1056.28	$2MPSO_3\ 8*50*0.25*10^6$
c150	1096.53	$2MPSO_3\ 8*50*0.25*10^5$
c199	1362.84	$2MPSO_3\ 8*50*0.25*10^6$
f71	278.65	$2MPSO_3\ 8*50*0.25*10^3$
f134	11755.58	$2MPSO_3\ 8*50*0.25*10^6$
tai75a	1682.90	$2MPSO_3\ 8*50*0.25*10^6$
tai75b	1391.74	$2MPSO_3\ 8*25*1.1*10^5$
tai75c	1406.27	$TabuSearch\ 25*30seconds$
tai75d	1342.26	$MEMSO\ 8*25*0.5*10^4$
tai100a	2146.53	$2MPSO_2\ 8*25*1.1*10^5$
tai100b	2022.13	$2MPSO_3\ 8*50*0.25*10^5$
tai100c	1446.10	$2MPSO_3\ 8*50*0.25*10^6$
tai100d	1685.53	$2MPSO_2\ 8*25*1.1*10^5$
tai150a	3253.77	$MEMSO\ 8*25*0.5*10^4$
tai150b	2861.91	$MAPSO\ 8*25*0.5*10^4$
tai150c	2462.96	$2MPSO_3\ 8*50*0.25*10^6$
tai150d	2844.70	$2MPSO_2\ 8*25*1.1*10^5$

tributed to their higher flexibility in space searching due to the possibility of efficient parallelization with low synchronization cost.

On a general note, we believe that the proposed method offers three main advantages: simplicity, multi-cluster characteristics, and partially "method-free" nature. The simplicity is attributed to the mechanism of knowledge transfer between swarms and between consecutive phases, which requires minimal memory involvement - only the best particle (solution) needs to be maintained and propagated. The multi-cluster aspect of the method is manifested by the possibility of assigning several requests' clusters per vehicle, which significantly increases the flexibility of possible space clustering, though at the cost of adequate increase of search space dimensionality. Finally, the universality of the method stems from the potential ability of using any global, continuous optimization method in the second phase of the algorithm. Our choice was to employ the PSO method at this stage, however, there are no apparent reasons for not using other candidate methods.

In the future we plan to develop a hyper-heuristic approach to the parameter/method selection problem. We have already started to explore this topic with promising initial results and believe that further studies may lead to development of robust and effective autonomous self-adaptation procedure. As can be observed in the example benchmarks' characteristics presented in Figs. 8, 9 and 10, spatial and volume distributions of requests significantly vary between test instances what may be taken into account in the parameter selection procedure.

We also plan to verify the efficacy of our method on the real-life data collected by a Warsaw-based delivery company.

ACKNOWLEDGMENT

The research was financed by the National Science Centre in Poland, grant number DEC-2012/07/B/ST6/01527 and by the research fellowship within "Information technologies: Research and their interdisciplinary applications" agreement number POKL.04.01.01-00-051/10-00.

REFERENCES

[1] G. B. Dantzig and R. Ramser., "The Truck Dispatching Problem," *Management Science*, vol. 6, pp. 80–91, 1959.

[2] J. K. Lenstra and A. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, pp. 221–227, 1981.

[3] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems." *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/eor/eor225.html\#PillacGGM13

[4] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E.-G. Talbi, "A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests," *Applied Soft Computing*, vol. 12, no. 4, pp. 1426–1439, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494611004339

[5] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981. [Online]. Available: http://dx.doi.org/10.1002/net.3230110205

[6] D. M. Ryan, C. Hjorring, and F. Glover, "Extensions of the petal method for vehicle routing," *J. Oper. Res. Soc.*, vol. 44, pp. 289–296, 1993.

[7] É. D. Taillard, "Parallel iterative search methods for vehicle routing problems," *Networks*, vol. 23, no. 8, pp. 661–673, 1993.

[8] P. Kilby, P. Prosser, and P. Shaw. (1998) Dynamic VRPs: A Study of Scenarios. [Online]. Available: http://www.cs.strath.ac.uk/~apes/apereports.html

[9] L. Meidan, S. Yehua, W. jing, and Z. Feifei, "Insertion heuristic algorithm for dynamic vehicle routing problem with time window," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, Dec 2010, pp. 3789–3792.

[10] M. Albareda-Sambola, E. Fernández, and G. Laporte, "The dynamic multiperiod vehicle routing problem with probabilistic information," *Computers & Operations Research*, vol. 48, pp. 31–39, 2014.

[11] L. Feng, Y.-S. Ong, I. W.-H. Tsang, and A.-H. Tan, "An evolutionary search paradigm that learns with past experiences," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[12] X. Chen, L. Feng, and Y. Ong, "A self-adaptive memeplexes robust search scheme for solving stochastic demands vehicle routing problem," *Int. J. Systems Science*, vol. 43, no. 7, pp. 1347–1366, 2012.

[13] F. T. Hanshar and B. M. Ombuki-Berman, "Dynamic vehicle routing using genetic algorithms," *Applied Intelligence*, vol. 27, no. 1, pp. 89–99, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1007/s10489-006-0033-z

[14] R. Montemanni, L. Gambardella, A. Rizzoli, and A. Donati, "A new algorithm for a dynamic vehicle routing problem based on ant colony system," *Journal of Combinatorial Optimization*, vol. 10, pp. 327–343, 2005.

[15] M. Okulewicz and J. Mańdziuk, "Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem," in *Artificial Intelligence and Soft Computing*, ser. Lecture Notes in Computer Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds., vol. 7895. Springer Berlin Heidelberg, 2013, pp. 547–558.

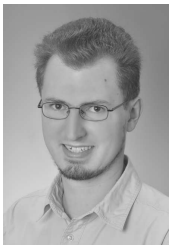
[16] ——, "Two-Phase Multi-Swarm PSO and the Dynamic Vehicle Routing Problem," in *2nd IEEE Symposium on Computational Intelligence for Human-like Intelligence*. Orlando, Fl, USA: IEEE Press, 2014, pp. 86–93.

[17] M. R. Khouadjia, E.-G. Talbi, L. Jourdan, B. Sarasola, and E. Alba, "Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems," *Journal of Supercomputing*, vol. 63, no. 3, pp. 836–853, 2013.

[18] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks. IV*, pp. 1942–1948, 1995.

[19] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," *Proceedings of Evolutionary Programming VII (EP98)*, pp. 591–600, 1998.

[20] ——, "A modified particle swarm optimizer," *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, 1998.
[21] I. Cristian and Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.
[22] M. Clerc. (2012) Standard PSO 2007 and 2011. [Online]. Available: http://www.particleswarm.info/
[23] C. K. Monson and K. D. Seppi, "Exposing origin-seeking bias in PSO," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 241–248. [Online]. Available: http://doi.acm.org/10.1145/1068009.1068045
[24] W. M. Spears, D. T. Green, and D. F. Spears, "Biases in particle swarm optimization," *International Journal of Swarm Intelligence Research*, vol. 1(2), pp. 34–57, 2010.
[25] C. Lin, K. Choy, G. Ho, H. Lam, G. K. Pang, and K. Chin, "A decision support system for optimizing dynamic courier routing operations," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6917–6933, 2014.
[26] M. R. Khouadjia, E. Alba, L. Jourdan, and E.-G. Talbi, "Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, 2010, vol. 6234, pp. 227–238. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15461-4_d20
[27] L. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *IEEE Conference on Evolutionary Computation*, 1996, pp. 622–627.
[28] I. H. Osman and N. Christofides, "Capacitated clustering problems by hybrid simulated annealing and tabu search," *International Transactions in Operational Research*, vol. 1, no. 3, pp. 317–336, 1994. [Online]. Available: http://dx.doi.org/10.1111/1475-3995.d01-43
[29] G. Croes, "A method for solving traveling salesman problems," *Operations Res. 6*, pp. 791–812, 1958.
[30] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, pp. 48–50, 1959.
[31] N. Christofides and J. E. Beasley, "The period routing problem," *Networks*, vol. 14, no. 2, pp. 237–256, 1984. [Online]. Available: http://dx.doi.org/10.1002/net.3230140205
[32] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981. [Online]. Available: http://dx.doi.org/10.1002/net.3230110205
[33] G. Pankratz and V. Krypczyk. (2009) Benchmark data sets for dynamic vehicle routing problems. [Online]. Available: http://www.fernuni-hagen.de/WINF/inhalte/benchmark_data.htm
[34] J. Mańdziuk, S. Zadrożny, K. Walędzik, M. Okulewicz, and M. Świechowski. (2015) Adaptive metaheuristic methods in dynamically changing environments. [Online]. Available: http://www.mini.pw.edu.pl/~mandziuk/dynamic/



Michał Okulewicz received his MSc degree in Computer Science from Warsaw University of Technology, Warsaw Poland in 2011. Since 2009 he has been working as a software developer and designer and since 2011 as a research assistant at the Warsaw University of Technology. He is currently working on the PhD project on Swarm Optimization in dynamic environments. He is interested in applications of Artificial Intelligence and a Geographical Information Systems enthusiast.

Jacek Mańdziuk Ph.D., D.Sc., received M.Sc. (Honors) and Ph.D. in Applied Mathematics from the Warsaw University of Technology, Poland in 1989 and 1993, resp., D.Sc. degree in Computer Science from the Polish Academy of Sciences in 2000 and the title of Professor Titular in 2011.

He is an Associate Professor at the Warsaw University of Technology and Head of the Division of Artificial Intelligence and Computational Methods. He is the author of three books (including *Knowledge-free and Learning-based Methods in Intelligent Game Playing, Springer-Verlag, 2010*) and over 100 peer-reviewed papers.

His current research interests include application of Computational Intelligence methods to game playing, dynamic optimization problems, bioinformatics, financial modeling and development of general-purpose human-like learning methods.

Prof. Mańdziuk is an Associate Editor of the IEEE Transactions on Computational Intelligence and AI in Games and an Editorial Board Member of the International Journal On Advances in Intelligent Systems. In 1996 he received the Fulbright Senior Advanced Research Award.