Two-Phase Multi-Swarm PSO and the Dynamic Vehicle Routing Problem

Michał Okulewicz and Jacek Mańdziuk

***Abstract*—In this paper a new 2-phase multi-swarm Particle Swarm Optimization approach to solving Dynamic Vehicle Routing Problem is proposed and compared with our previous single-swarm approach and with the PSO-based method proposed by other authors. Furthermore, several evaluation functions and problem encodings are proposed and experimentally verified on a set of standard benchmark sets. For a cut-off time set in the middle of a day our method found new best-literature results for 17 out of 21 tested problem instances.**

I. Introduction

Dynamic Vehicle Routing Problem (DVRP) is a hard combinatorial optimization problem. It is a generalization of the Traveling Salesman Problem (TSP) with additional travel time and vehicle capacity constrains. In the problem instances considered in this paper new clients' requests may arrive during the whole working day.

The problem is demanding for both humans and machines. When solving the DVRP people heavily rely on their life experience, imagination and the ability to develop geometry-based graphical solutions. While life experience and imagination are, to a large extent, beyond the scope of current machines' capabilities, the ability to move in a "geometrically-guided" way in the search space in order to detect the optimal cluster centers for individual vehicles' routes can be, apparently quite effectively, accomplished by artificial agents. One of such possibilities, based on the PSO meta-heuristic is proposed in this paper.

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 and the TSP for finding an optimal route for a given vehicle. Such a combination may be effectively solved by approximate or metaheuristic algorithms, e.g. [1],[2], [3]. The solution method chosen by the authors consists in applying Particle Swarm Optimization (PSO) algorithm to solving both of these sub-problems [4]. To the best of our knowledge it is the first published attempt of applying a two-phase PSO approach to solving the DVRP[1].

In this paper, we significantly improve our previous approach by introducing a new multi-swarm algorithm, which is tested against various fitness functions and problem encodings. We compare our results with the state-of-the-art solutions and for a cut-off time equal to 0.5 (i.e. set in the middle of a day) we present new best results in the case of 17 out of 21 benchmark problems tested in our experiments.

The rest of this paper is organized as follows. In section II the PSO algorithm and its parameters are briefly presented. In section III mathematical model for the DVRP is given. In the subsequent sections (IV, V and VI) various problem encodings and fitness functions for the PSO method are proposed and discussed. Finally, sections VII and VIII present the results and conclusions.

II. Particle Swarm Optimization Algorithm

PSO is an iterative global optimization metaheuristic method proposed in 1995 by Kennedy and Eberhart [5] and further studied and developed by many other researchers, e.g., [6], [7], [8]. In short, PSO utilizes the idea of swarm intelligence to solve hard optimization tasks.

The basic idea of the PSO algorithm consists in maintaining a swarm of particles moving in the search space. Particles which communicate (to a given particle) their position and function value in that position are called neighbours of that particle. Each particle maintains its current position, velocity and a set of neighbours, as well as remembers its best visited location.

Update position: In every step t, position of particle i, x_t^i is updated based on particle's velocity w_t^i:

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

Update velocity: In our implementation of PSO (based on [9] and [6]) velocity w_t^i of particle i is updated according to the following rule:

$$\begin{aligned} w_{t+1}^i =& u_{U[0;g]}^{(1)}(x_{best}^{neighbours_i} - x_t^i) + \\ & u_{U[0;l]}^{(2)}(x_{best}^i - x_t^i) + \\ & a w_t^i, \end{aligned} \tag{2}$$

where

- g is a neighbourhood attraction factor,
- x_{best}^i and $x_{best}^{neighbours_i}$ represent the best position (in terms of solving the problem) found hitherto, respectively by particle i and by the neighbourhood of the ith particle,
- l is a local attraction factor,
- 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.

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).

[1]Division of a solution method into clustering and routes' optimization phases was previously proposed in several methods that were rooted in the Operational Research fields.

III. DVRP Definition

In the class of Dynamic Vehicle Routing Problems discussed in this article one considers a fleet V of n vehicles and a series C of m clients (requests) to be served (a cargo is to be delivered to them).

The fleet of vehicles is homogeneous. Vehicles have identical $capacity \in \mathbb{R}$ and the same $speed$ [2] $\in \mathbb{R}$.

The cargo is taken from a dedicated depot d which has a certain $location_0 \in \mathbb{R}^2$ and working hours $(start, end)$, where $0 \leq start < end$.

Each client $c_l, l = 1, \ldots, m$ 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, and $size_l \in \mathbb{R}$ - size of the request ($size_l \leq capacity$).

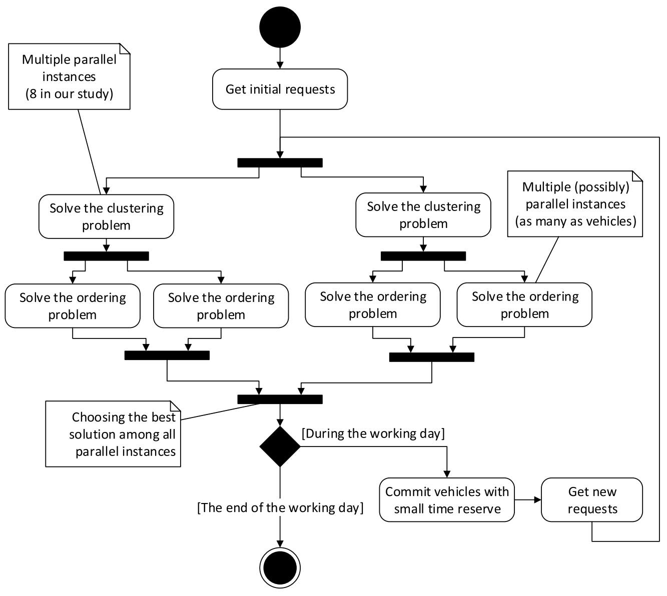
A travel distance $\rho(i, j)$ is the Euclidean distance between $location_i$ and $location_j$ on the $\mathbb{R}^2$ plane, $i, j = 0, \ldots, m$.

The $route_i$ of vehicle v_i is a series of p_i locations, where $location_{i_1}$ and $location_{i_{p_i}}$ are locations of a depot and a series of p_i time points of arrivals at those locations (denoted arv_r for $location_r$).

As previously stated, the goal is to serve the clients (requests), according to their defined constraints, with minimal total cost (travel distance). Formally, the optimization goal and constraints can be written as:

$$
\begin{gathered}
\min \sum_{i=1}^{n} \sum_{r=1}^{p_i} \rho(i_{r-1}, i_r) \\
\forall_{i \in [n]} \forall_{r \in [p_i] / \ \{1\}} arv_{i_r} \geq arv_{i_{r-1}} + \rho(i_{r-1}, i_r) + unld_{r_{i-1}} \\
\forall_{i \in [n]} arv_{i_1} \geq start_{i_1} \\
\forall_{i \in [n]} arv_{i_p} \leq endi_p \\
\forall_{i \in [n]} \sum_{r=2}^{p_i - 1} size_r \leq capacity \\
\forall_{l \in \{1,2,\ldots m\}} \exists!_{i \in [n]} location_{l+k} \in route_i
\end{gathered}
\tag{3}
$$

Please note that, according to equation (eq. 3), each client must be assigned to exactly one vehicle and all vehicles must return to the depot before its closing.



Multiple parallel instances (8 in our study)
Get initial requests
Solve the clustering problem
Solve the clustering problem
Multiple (possibly) parallel instances (as many as vehicles)
Solve the ordering problem
Solve the ordering problem
Solve the ordering problem
Solve the ordering problem
Choosing the best solution among all parallel instances
[During the working day]
Commit vehicles with small time reserve
Get new requests
[The end of the working day]

Fig. 1. Activity diagram of the multi-swarm 2PSO algorithm

TABLE I

Comparison of baseline algorithms: 2PSO (a single swarm approach proposed in [4]); 2MPSOv1 (a multi swarm version of this method); and MAPSO proposed by Khouadjia et al. [12]. All methods are compared on the common ground of 10^4 calculations of the PSO evaluation function per time slice.

	2MPSOv1 (10^4)		2PSO (10^4)		MAPSO (10^4)	
	Min	Avg	Min	Avg	Min	Avg
c50	589.24	626.57	582.88	675.14	**571.34**	**610.67**
c75	937.41	988.58	**912.23**	1015.16	931.59	**965.53**
c100	957.54	1041.18	996.4	1149.48	**953.79**	**973.01**
c100b	**828.94**	**849.69**	**828.94**	*850.68*	866.42	882.39
c120	**1079.14**	**1162.65**	1087.04	1212.38	1223.49	1295.79
c150	**1173.25**	**1240.9**	1173.94	1336.84	1300.43	1357.71
c199	**1408.74**	**1458.24**	1446.93	1578.99	1595.97	1646.37
f71	293.1	317.15	315	356.75	**287.51**	**296.76**
f134	**12304.03**	**12587.49**	12813.14	13491.6	15150.5	16193
tai75a	1814.95	1958.95	1871.06	2142.07	**1794.38**	**1849.37**
tai75b	1435.76	1486.39	1460.95	1568.21	**1396.42**	**1426.67**
tai75c	1497.64	1660.36	1500.23	1811.08	**1483.1**	**1518.65**
tai75d	1459.68	1496.22	1462.82	1586.28	**1391.99**	**1413.83**
tai100a	2198.02	2381.24	2317.76	2707.61	**2178.86**	**2214.61**
tai100b	**2134.31**	2267.1	2187.86	2510.6	2140.57	**2218.58**
tai100c	1555.73	1611.17	1564.25	1672.33	**1490.4**	**1550.63**
tai100d	**1819.56**	1939.34	1859.7	2220.01	1838.75	**1928.69**
tai150a	3480.84	3667.11	3638.75	4151.31	**3273.24**	**3389.97**
tai150b	3004.98	3118	3107.95	3302.94	**2861.91**	**2956.84**
tai150c	2714.25	2821.53	2781.02	2952.88	**2512.01**	**2671.35**
tai150d	3029.75	3174.42	3048.24	3478.49	**2861.46**	**2989.24**
sum	**45716.86**	**47854.28**	46957.09	51770.83	48104.13	50349.66

IV. Solving the DVRP

There are two general approaches to solving dynamic optimization problems. In the first one the optimization algorithm is run every time there is a change in the problem instance. In the second approach time is divided into discrete slices and the algorithm is run once for each time slice. Furthermore, the problem instance is considered ”frozen” during the whole time slice, i.e. any potential changes introduced during the current time slot are handled in the next algorithm's run (in the subsequent time slice period).

In our study we follow the second approach which, in the context of the DVRP, was proposed by Kilby et al. [10]. In order to assure a direct comparison of obtained results with our previous work [4] and with other PSO-based approaches [11], [12], the number of time slices of the working day is equal to 25.

Another critical DVRP parameter, which has a direct impact on “the degree of dynamism” of a given problem instance, is the *cut off time* which defines the part of requests that is known at the beginning of the working day. In real (practical) situations the requests received after this time threshold are treated as received at the beginning of the subsequent working day. In the one-day-horizon simulations presented in this paper (as well as in practically all other papers referring to Kilby et al.'s benchmarks [13]) the requests located after the cut off time limit are simply treated

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

TABLE III
MULTI-SWARM ALGORITHMS COMPARISON. ALL METHODS ARE COMPARED ON THE COMMON GROUND OF 10^4 CALCULATIONS OF THE PSO EVALUATION FUNCTION PER TIME SLICE.

	2MPSOv1 (10^4)		2MPSOv2 (10^4)		2MPSOv3 (10^4)		MAPSO (10^4)	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	589.24	626.57	578.61	**608.71**	583.09	*618.59*	**571.34**	*610.67*
c75	937.41	988.58	**892.04**	**944.18**	904.83	*946.85*	931.59	965.53
c100	957.54	1041.18	941.3	*969.16*	**926.1**	**966.27**	953.79	*973.01*
c100b	**828.94**	**849.69**	833.25	875.92	830.58	875.47	866.42	882.39
c120	1079.14	**1162.65**	**1061.01**	*1178.43*	1061.84	*1176.38*	1223.49	1295.79
c150	1173.25	1240.9	1142.57	**1196.88**	**1132.12**	*1208.6*	1300.43	1357.71
c199	1408.74	*1458.24*	1394.61	*1470.16*	**1371.61**	**1458.01**	1595.97	1646.37
f71	293.1	317.15	291.2	313.92	302.5	319.01	**287.51**	**296.76**
f134	12304.03	12587.49	12011.71	*12509.83*	**11944.86**	**12416.65**	15150.5	16193
tai75a	1814.95	1958.95	1742.31	*1869.48*	**1721.81**	**1846.03**	1794.38	*1849.37*
tai75b	1435.76	1486.39	1401.22	1470.92	1418.82	1451.92	**1396.42**	**1426.67**
tai75c	1497.64	1660.36	1461.74	1547.57	**1456.9**	1560.68	1483.1	**1518.65**
tai75d	1459.68	1496.22	1421.48	1463.29	1445.58	1481.25	**1391.99**	**1413.83**
tai100a	2198.02	2381.24	2197.94	2278.07	2211.3	2327.2	**2178.86**	**2214.61**
tai100b	2134.31	2267.1	**2045.47**	*2156.24*	2052.54	**2131.91**	2140.57	2218.58
tai100c	1555.73	1611.17	1480.89	1541.56	**1465.06**	**1519.44**	1490.4	1550.63
tai100d	1819.56	1939.34	1739.25	**1789.74**	**1722.16**	*1808.67*	1838.75	1928.69
tai150a	3480.84	3667.11	3350.14	3527.45	3367.55	3537.81	**3273.24**	**3389.97**
tai150b	3004.98	3118	2918.39	3032.84	2911.22	3033.83	**2861.91**	**2956.84**
tai150c	2714.25	2821.53	**2497.55**	*2603.02*	2510.51	**2579.72**	2512.01	2671.35
tai150d	3029.75	3174.42	2915.98	3000.5	2893.54	2992.53	**2861.46**	**2989.24**
sum	45716.86	47854.28	44318.66	46347.87	**44234.52**	**46256.82**	48104.13	50349.66

TABLE II
BEST OVERALL RESULTS ACHIEVED FOR THE BENCHMARK SETS. THE RIGHTMOST COLUMN PRESENTS THE PARTICULAR ALGORITHM AND THE NUMBER OF THE EVALUATION FUNCTION CALCULATIONS PER TIME SLICE IN THE WINNING CASE.

Name	**Best result**	**Algorithm**
c50	568.82	2MPSOv2 (10^3)
c75	892.04	2MPSOv2 (10^4)
c100	918.61	2MPSOv3 (10^5)
c100b	823.19	2MPSOv3 (10^5)
c120	1057.94	2MPSOv2 (10^5)
c150	1121.50	2MPSOv2 (10^5)
c199	1371.61	2MPSOv3 (10^4)
f71	287.51	MAPSO (10^4)
f134	11944.86	2MPSOv3 (10^4)
tai75a	1721.81	2MPSOv3 (10^4)
tai75b	1391.74	2MPSOv3 (10^5)
tai75c	1440.20	2MPSOv2 (10^5)
tai75d	1391.99	MAPSO (10^4)
tai100a	2146.53	2MPSOv2 (10^5)
tai100b	2039.31	2MPSOv3 (10^5)
tai100c	1463.83	2MPSOv3 (10^5)
tai100d	1685.53	2MPSOv2 (10^5)
tai150a	3273.24	MAPSO (10^4)
tai150b	2861.91	MAPSO (10^4)
tai150c	2472.70	2MPSOv2 (10^5)
tai150d	2844.70	2MPSOv2 (10^5)

as being known at the beginning of the current day - they compose an initial instance of the DVRP being solved.

In the experiment presented in the paper, in order to allow a direct comparison with previous works, the cut-off time is set in the middle of a depot's working hours, i.e. lasts for half of a day.

V. PROBLEM ENCODING

Due to natural graph-based DVRP representation, various problem encodings have been tested in the literature. In particular, Khouadjia et al. proposed Dynamic Adapted PSO (DAPSO) (and its multi-swarm equivalent Multiswarm Adaptive Memory PSO (MAPSO)) [11], [12] which uses a discrete version of PSO to solve the DVRP. In our approach, denoted by 2-Phase Particle Swarm Optimization (2PSO) [4] a truly continuous problem encoding is proposed. Furthermore, we propose splitting the process of solving the 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 is found with the use of (a separate instance of) the standard PSO algorithm. In this paper we present three versions of this method differing mainly by the fitness functions used.

Additionally, we introduce a 2-Phase Multi-Swarm PSO (2MPSO) algorithm for each of the proposed versions of the 2PSO algorithm. The main technical difference between single- and multi- swarm versions is a need for synchronization of problems between swarms in the latter case, discussed in section VI. A flow-chart of the above described multi-swarm 2PSO algorithm is presented in Fig. 1. In all three versions of the algorithm which are discussed below clients (requests) are assigned to vehicles whose centers are the

TABLE IV
COMPARISON OF PROPOSED MULTI-SWARM ALGORITHM WITH ONE CLUSTER OF REQUESTS PER VEHICLE (2MPSOv2) AND VARIOUS NUMBERS OF FITNESS EVALUATIONS PER TIME SLICE WITH THE MAPSO METHOD WITH THE REFERENCED NUMBER OF THE FITNESS FUNCTION CALCULATIONS PER TIME SLICE (10^4).

	2MPSOv2 (10^3)		2MPSOv2 (10^4)		2MPSOv2 (10^5)		MAPSO (10^4)	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	**568.82**	629.08	578.61	**608.71**	571.53	*614.61*	571.34	*610.67*
c75	908.6	972.58	**892.04**	*944.18*	896.33	**930.94**	931.59	965.53
c100	982	1033.03	941.3	*969.16*	**920.11**	**957.49**	953.79	973.01
c100b	**829.54**	898.19	833.25	**875.92**	848.5	*881.7*	866.42	*882.39*
c120	1060.39	**1158.68**	1061.01	*1178.43*	**1057.94**	*1174.65*	1223.49	1295.79
c150	1138.03	1269.34	1142.57	1196.88	**1121.5**	**1168.59**	1300.43	1357.71
c199	1447.55	1581.75	**1394.61**	*1470.16*	1404.46	**1461.58**	1595.97	1646.37
f71	300.46	333.37	291.2	313.92	302.5	317.66	**287.51**	**296.76**
f134	12079.07	12529.43	12011.71	12509.83	**11988.76**	**12324.98**	15150.5	16193
tai75a	1798.99	1993.76	1742.31	1869.48	**1727.89**	**1812.55**	1794.38	1849.37
tai75b	1441.3	1530.25	1401.22	1470.92	1400.33	1438.5	**1396.42**	**1426.67**
tai75c	1511.04	1616.08	1461.74	1547.57	**1440.2**	**1491.64**	1483.1	*1518.65*
tai75d	1432.06	1502.44	1421.48	1463.29	1439.27	1470.93	**1391.99**	**1413.83**
tai100a	2244.3	2413.5	2197.94	2278.07	**2146.53**	2260.21	2178.86	**2214.61**
tai100b	2073.27	2247.82	2045.47	*2156.24*	**2045.24**	**2119.36**	2140.57	2218.58
tai100c	1527.25	1598.37	1480.89	1541.56	**1469.12**	**1516.97**	1490.4	1550.63
tai100d	1762.17	1845.61	1739.25	*1789.74*	**1685.53**	**1775.09**	1838.75	1928.69
tai150a	3566.83	3898.32	3350.14	3527.45	3345.88	3402.23	**3273.24**	**3389.97**
tai150b	3005.9	3215.82	2918.39	3032.84	2885.21	**2942.49**	**2861.91**	*2956.84*
tai150c	2593.7	2727.12	2497.55	2603.02	**2472.7**	**2543.47**	2512.01	2671.35
tai150d	3042.33	3198.31	2915.98	3000.5	**2844.7**	**2949.2**	2861.46	2989.24
sum	45313.6	48192.85	44318.66	46347.87	**44014.23**	**45554.84**	48104.13	50349.66

closest ones (in terms of Euclidan distance).

A. 2(M)PSOv1

In the first version of our 2PSO approach [4] the following problem representation was used:

- In the first phase, each particle represents the centers of clusters of requests assigned to vehicles.
- The fitness function value in this phase is calculated as a sum of distances from the inter-cluster requests to the clusters' centers (a measure of quality of a clustering) and twice the distances from the clusters' centers to the depot location (a measure of a cost of creating a cluster).
- In the second phase, which is solved by a another (separate) multi-swarm approach each particle represents an ordering of requests assigned to a given vehicle (each cluster/vehicle is solved by a separate PSO instance).
- The fitness value in this 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 value is the sum of fitness functions' values of the best solutions found by each of the PSO instances.

B. 2(M)PSOv2

The second version of the 2PSO algorithm differs from the above-described basic variant, by the fitness function used in the first phase. Here, the estimated total length of all vehicles' routes defined by the proposed clusters and optimized with the help of 2-OPT, is used.

The cost functions calculated in phase 1 (requests clustering phase) for all three versions are schematically presented in Fig. 2.

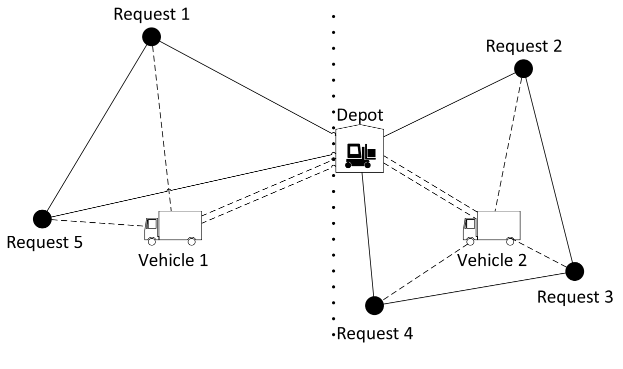


Fig. 2. Example of a DVRP problem with 2 vehicles and 5 clients' requests. Solid lines represent possible routes, whose lengths are used as an evaluation function by 2(M)PSOv2 and 2(M)PSOv3. Dashed lines represent estimated cluster cost (quality), which is used as an evaluation function by 2(M)PSOv1. Each vehicle is located in the respective cluster's center. Dotted line separates the two operating areas assigned to vehicles.

C. 2(M)PSOv3

The third version of the 2PSO algorithm uses the same fitness function as above in the first phase but allows for

TABLE V
COMPARISON OF PROPOSED MULTI-SWARM ALGORITHM WITH 3 CLUSTERS OF REQUESTS PER VEHICLE (2MPSOV3) AND VARIOUS NUMBERS OF FITNESS EVALUATIONS PER TIME SLICE WITH THE MAPSO METHOD WITH THE REFERENCED NUMBER OF THE FITNESS FUNCTION CALCULATIONS PER TIME SLICE (10^4).

	2MPSOv3 (10^3)		2MPSOv3 (10^4)		2MPSOv3 (10^5)		MAPSO (10^4)	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	603.61	632.05	583.09	618.59	571.53	610.89	**571.34**	**610.67**
c75	917.51	965.06	**904.83**	*946.85*	906.72	**933.4**	931.59	965.53
c100	949.9	999.71	926.1	*966.27*	**918.61**	**953.13**	953.79	973.01
c100b	835.71	*886.21*	830.58	*875.47*	**823.19**	**871.07**	866.42	*882.39*
c120	1076.15	**1170.64**	**1061.84**	*1176.38*	1112.73	*1174.04*	1223.49	1295.79
c150	1178.36	1240.35	**1132.12**	*1208.6*	1150.13	**1187.27**	1300.43	1357.71
c199	1440.99	1528.83	**1371.61**	**1458.01**	1427.73	*1487.29*	1595.97	1646.37
f71	302.5	327.09	302.5	319.01	311.59	319.23	**287.51**	**296.76**
f134	12015.9	*12403.53*	**11944.86**	*12416.65*	12022.64	**12312.74**	15150.5	16193
tai75a	1749.65	1935.95	**1721.81**	*1846.03*	1760.27	**1821.01**	1794.38	1849.37
tai75b	1416.61	1513.96	1418.82	1451.92	**1391.74**	**1426.39**	1396.42	*1426.67*
tai75c	1450.55	*1575.28*	1456.9	1560.68	**1446.85**	1529.36	1483.1	**1518.65**
tai75d	1437.93	1495.25	1445.58	1481.25	1435.92	*1465.55*	**1391.99**	**1413.83**
tai100a	2268.83	2419.43	2211.3	2327.2	**2169.24**	2248.91	2178.86	**2214.61**
tai100b	2087.39	2232.06	2052.54	*2131.91*	**2039.31**	**2094.49**	2140.57	2218.58
tai100c	1520.35	1581.29	1465.06	*1519.44*	**1463.83**	**1516.35**	1490.4	*1550.63*
tai100d	1731.95	*1829.63*	1722.16	*1808.67*	**1690.89**	**1778.74**	1838.75	1928.69
tai150a	3594.95	3841.77	3367.55	3537.81	3319.48	3430.9	**3273.24**	**3389.97**
tai150b	3001.21	3196.83	2911.22	3033.83	2901.2	2973.39	**2861.91**	**2956.84**
tai150c	2547.41	2699.46	2510.51	2579.72	**2483.39**	**2525.04**	2512.01	2671.35
tai150d	2965.78	3148.84	2893.54	*2992.53*	2868.94	**2956.74**	**2861.46**	*2989.24*
sum	45093.24	47623.22	44234.52	46256.82	**44215.93**	**45615.93**	48104.13	50349.66

assignment of up to $c > 1$ requests clusters to the same vehicle. This allows formation of the routes which are composed of up to c distinct subroutes (fragments), which otherwise, due to the proximity assignment rule mentioned above, would not be possible.

If we denote by m the number of requests ($50 \leq m \leq 199$ in the benchmarks used), by n the number of available vehicles ($n = 50$ in the tested benchmarks), by $\hat{n}$ the estimated number of required vehicles (usually around $\frac{n}{3}$), by c the number of clusters for each of the vehicles (we used $c = 1$ or $c = 3$ in our study) and by b ($b = 4$ was used in our test) the number of spare vehicles (beyond the estimated need - for the safety of the method), then the theoretical and the experimental dimension sizes of the search spaces are as follows:

Algorithm	Theoretical size	Experimental size
2(M)PSOv1	$2n$	100
2(M)PSOv2	$2n$	100
2(M)PSOv3	$2c(\hat{n} + b)$	$[60, 120]$

VI. KNOWLEDGE TRANSFER

In dynamic problems, one of the crucial tasks is proper and 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 issue is the problem of efficient usage of parallel or distributed architecture and knowledge transfer between partial problems within the same problem instance (time slot).

A. *Knowledge transfer between time slices*

In the MAPSO algorithms Khouadjia et al. proposed adding an adaptive memory to each particle, in order to store its best known solution (the vector of vehicles identifiers which are assigned to each of the 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 algorithm.

In the 2(M)PSO method 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.

B. *Knowledge transfer between swarms*

In the MAPSO algorithm knowledge is transferred between swarms by migrating particles. In every iteration for each of the particles there is a small probability that a particle will migrate to a different swarm. As MAPSO allows for distributed way of solving the problem there is, in general, no guarantee that in a given moment all swarms are solving the problem for 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 multithreading computer. Therefore, we take an easy approach where, 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 again among the threads (cf. Fig. 1). Such approach is motivated by the assumption that at the end of a time slice some vehicles are committed to serve a given set of requests and it might be meaningless to solve problem instances not synchronized with the current state of the problem instance.

VII. Results

In order to evaluate the performance of the algorithm we used dynamic versions of Chritofides', Fisher's and Taillard's benchmark sets [13]. We compare our algorithm with the MAPSO approach which provided the best so far average literature results and majority of the best known literature solutions (minima). The basic comparison is made for the same number of swarms and the same number of fitness function evaluations per time slice. Additionally we present the results for higher numbers of function evaluations per time slice to check whether using more function evaluations will further improve the 2(M)PSO results. Due to inability to precisely replicate the method, results for MAPSO are presented only in the reference case of 10^4 function evaluations per time slice published by Khouadija et al.

In the experiments, parameters of the PSO method were set in the following way:

Parameter	Value
neighbourhood attraction factor (g)	0.60
local attraction factor (l)	2.20
inertia coefficient (a)	0.63
$P(X$ is a neighbour of $Y)$	0.50
#iterations	$\{50, 250, 1000\}$
#particles	$\{20, 40, 100\}$
#swarms	$\{1, 8\}$

$P(X$ is a neighbour of $Y)$ is a probability that particle X belongs to the neighbourhood of particle Y. Note that the relation of being a neighbor is not symmetrical.

Figure 3a presents the aggregated minimum and average values together with standard error bars obtained for the basic single-swarm (2PSO) and multi-swarm (2MPSO) versions of the 2PSO algorithm.

Figures 3b, 4a and 4b present the same type of data, respectively for various multi-swarm versions of the 2MPSO algorithm, the 2MPSOv2 algorithm with various numbers of fitness function evaluations, and the 2MPSOv3 algorithm with various numbers of fitness function evaluations.

In each plot a solid horizontal line marks the average performance of the MAPSO algorithm for the problem instances in a given benchmark set, while dashed horizontal line depicts the average of the best results of MAPSO for a given benchmark set.

A detailed comparison of numerical results for the same experiments is presented in Tables I, III, IV and V, respectively with the best results marked in bold and statistically insignificantly worse average results marked in italics. The significance of the differences in results between the 2PSO/2MPSO methods was calculated with the use of the *Mann-Whitney U test* [16] and the significance of the differences in results between 2PSO and MAPSO was tested using the *Wilcoxon signed-rank test* [17] with a null hypothesis saying that a distribution of the 2PSO algorithm's results was symmetric around the average performance of MAPSO.

The best obtained results for each of the benchmark problems and the name and version of the algorithm which found them are listed in Table II.

VIII. Discussion and conclusions

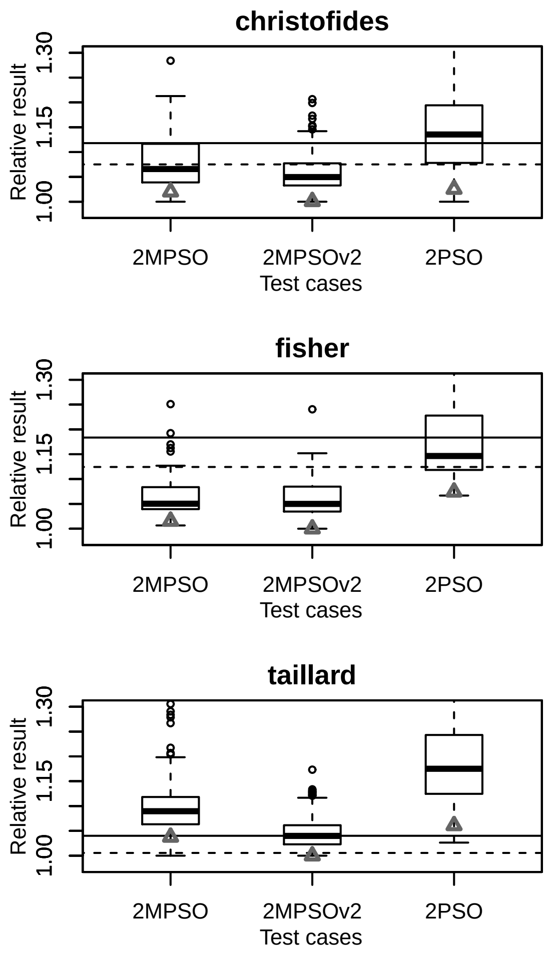
The highest performance of the 2PSO algorithm was accomplished using a multi-swarm version of this method. Even a simple collection of isolated swarms synchronized only once per time slice (at the end of it) allowed for visible gain in the solutions' quality (around 8%, on average).

Moreover, in most of the test cases, changing the fitness function from a total sum of clusters' weights (used by 2MPSOv1) to the estimation of a total sum of routes' lengths (used by 2MPSOv2 and 2MPSOv3) proved to be beneficial. The 2MPSO algorithms based on the modified, routes lengths-based fitness function used for optimizing the assignment, perform significantly better than the first version of the algorithm except for the cases of highly spatial clusters composed of uniformly distributed request sizes (e.g. instances *c100b*, *c120*, *c199*).

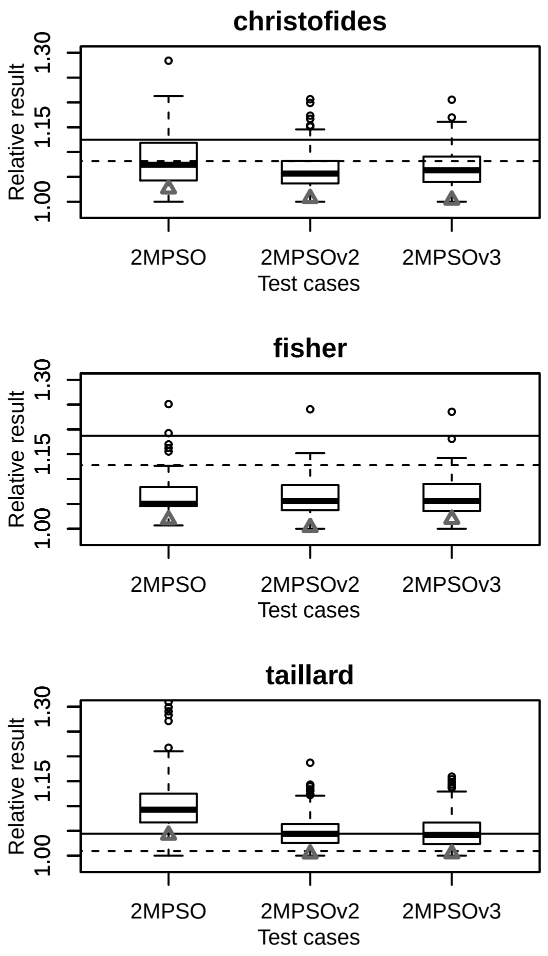
According to our intuition, changing the problem encoding from one cluster per vehicle to several clusters per vehicle should enhance the range of possible solutions (by directly allowing the overlaps among vehicles' routes). Even though some improvement is observable it is not statistically significant in terms of average results.

For the Christofides benchmark set the results get slightly deteriorated when the number of fitness function evaluations per time slice was raised to 10^5. This phenomenon may possibly stem from the fact, that the algorithm is stuck in the close-to-optimal solution for the initial time slices, which is not the part of the final optimal solution and, as a result, vehicles are committed too early to some of the requests. Detection of such "over-fitting" may potentially be used as a stopping criterion for the method.

During the experiments new best solutions for the cut-off time set to 0.5 (the same cut-off time was used by the referenced MAPSO-related works) were found for 17 out of 21 benchmark sets. All tested multi-swarm approaches outperformed MAPSO in terms of the average value of the



(a) Comparison for all the benchmark sets for the baseline versions of the algorithms



(b) Comparison for all the benchmark sets for the multi-swarm versions of the algorithms

Fig. 3. Comparison of performance of different 2PSO algorithm versions. Solid and dashed lines represent the means of average and best performance of MAPSO algorithm. A gray triangle represents the mean best performance for the given test case set.

best results and nearly all of them were more effective when the average values of (all) the results were considered (only 2MPSOv2 with $1\,000$ fitness evaluations per time slice found the routes which were, on average, 1% longer than those of MAPSO).

The best average results were achieved by 2MPSOv2 with 100 000 fitness evaluations per time slice (they were 3% better than MAPSO in terms of the average of the minima and nearly 5% better in terms of the average lengths) and 2MPSOv3 with 10 000 fitness evaluations per time slice (which outperformed MAPSO by nearly 4% in each of both above-mentioned efficiency measures).

The results suggest that problem instances could be differentiated based on the spatial distribution of request' locations, as well as on the requests' sizes distribution. The task of autonomous selection of problem encoding and fitness function to be used for a particular problem instance is one of our future research goals. We also plan to perform additional tests in order to further validate the 2(M)PSO ability to effectively solve the DVRP.

Acknowledgment

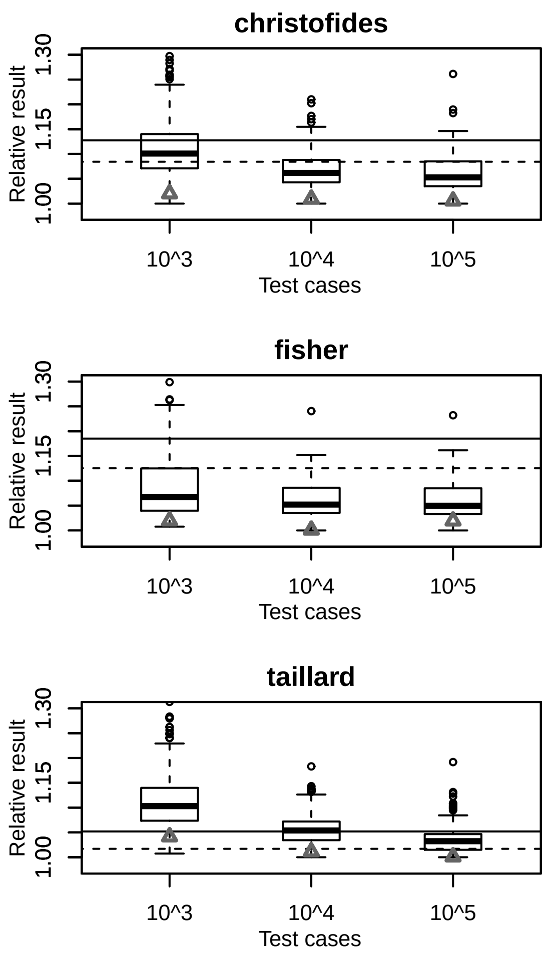
The research was financed by the National Science Centre in Poland grant number DEC-2012/07/B/ST6/01527 and by the research fellowship within "Information technologies: Research and their interdisciplinary applications" agreement number POKL.04.01.01-00-051/10-00.

References

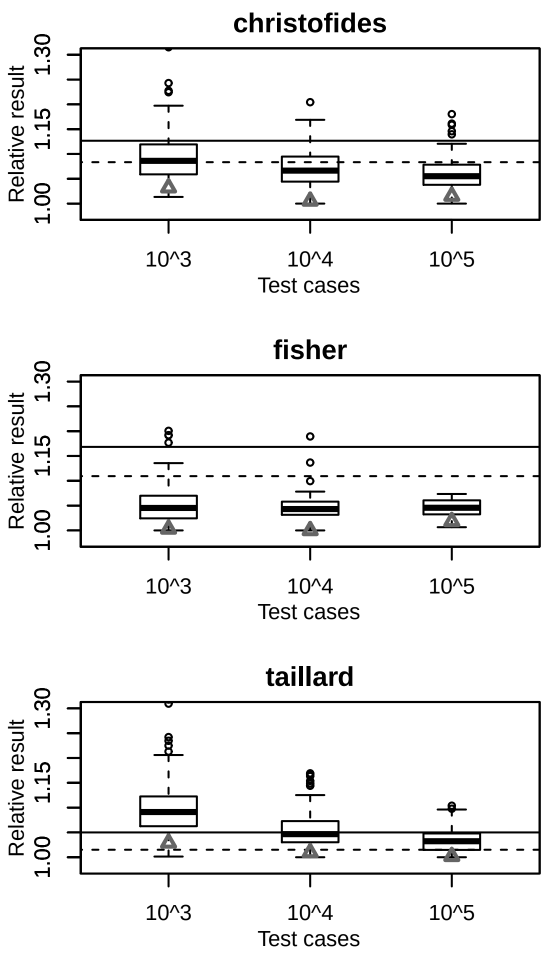
[1] L. Feng, Y.-S. Ong, I. W.-H. Tsang, and A.-H. Tan, "An evolutionary search paradigm that learns with past experiences," in *IEEE Congress on Evol. Comp.* IEEE, 2012, pp. 1–8.

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

[3] 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, p. 327343, 2005.



(a) Comparison for all the benchmark sets for the multi-swarm version with 1 cluster per vehicle and various numbers of function evaluations per time slice



(b) Comparison for all the benchmark sets for the multi-swarm version with 3 clusters per vehicle and various numbers of function evaluations per time slice

Fig. 4. Comparison of performance of multi-swarm 2MPSO algorithms for different number of fitness function evaluation. Solid and dashed lines represent the means of average and best performance of the MAPSO algorithm. A gray triangle represents the mean best performance for the given test case set.

[4] M. Okulewicz and J. Mańdziuk, "Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem," in *Artificial Intelligence and Soft Computing*, ser. LNCS, L. e. Rutkowski, Ed., vol. 7895. Springer Berlin Heidelberg, 2013, pp. 547–558.

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

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

[7] ——, "A modified particle swarm optimizer," *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, 1998.

[8] 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.

[9] M. Clerc, "Standard PSO 2007 and 2011," 2012. [Online]. Available: http://www.particleswarm.info/

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

[11] 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.

[12] M. 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. LNCS. Berlin / Heidelberg: Springer, 2010, vol. 6234, pp. 227–238. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15461-4_d20

[13] G. Pankratz and V. Krypczyk, "Benchmark data sets for dynamic vehicle routing problems," 2009. [Online]. Available: http://www.fernuni-hagen.de/WINF/inhalte/benchmark_data.htm

[14] G. Croes, "A method for solving traveling salesman problems," *Op. Res. 6*, pp. 791–812, 1958.

[15] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, p. 4850, 1959.

[16] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, p. 5060, 1947.

[17] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometr. Bull.*, vol. 1, p. 8083, 1945.