Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem

Michał Okulewicz and Jacek Mańdziuk

Warsaw University of Technology,
Faculty of Mathematics and Information Science,
Koszykowa 75, 00-662 Warsaw, Poland
{M.Okulewicz, J.Mandziuk}@mini.pw.edu.pl

Abstract. In this paper we apply Particle Swarm Optimization (PSO) algorithm to Dynamic Vehicle Routing Problem (DVRP) for solving the client assignment problem and optimizing the routes. Our approach to solving the DVRP consists of two stages. First we apply an instance of PSO to finding clients' requests that should be satisfied by the same vehicle (thus solving the Bin Packing Problem (BPP) part of the DVRP task) and then we apply several other instances of PSO, one for each of the vehicles assigned to particular requests (thus solving the Travelling Salesman Problem (TSP)). Proposed algorithm found better solutions then other tested PSO approaches in 6 out of 21 benchmarks and better average solutions in 5 of them.

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

1 Introduction

The goal of Vehicle Routing Problem (VRP) is to find the shortest routes for n homogeneous vehicles delivering cargo for m clients. Every client has its own demand of cargo. The capacity of each vehicle is limited and the cargo could be loaded on the vehicle on one of the k depots.

VRP could be regarded as a composition of two NPC problems: the Bin Packing Problem (optimizing the number of used vehicles by assigning client's requests to the vehicles) and the Travelling Salesman Problem (optimizing the route for each of the vehicles). In this study we are focused on the dynamic version of the problem (DVRP) where clients' requests (load demands and required destinations) are not fully available beforehand and are partly defined during the working day (the so-called Vehicle Routing Problem with Dynamic Requests [11]). The goal of DVRP is to find the shortest routes within the time bounds of the working day (i.e. the opening hours of the depots).

In recent years, biologically inspired computational intelligence algorithms have grown a lot of popularity. Examples of such algorithms, based on the idea

```
PSO() {
  swarm.initializeRandomlyParticlesLocationAndVelocity();
  for i from 1 to maxIterations {
    for each particle in swarm {
      particle.updateVelocity();
      particle.updateLocation();
    }
  }
}

Particle {
  updateVelocity() {
    for (i from 1 to dimensions) {
      this.v[i] =
        random.uniform(0,g)*(this.neighbours.best[i] - this.x[i]) +
        random.uniform(0,l)*(this.best[i] - this.x[i]) +
        random.uniform(0,r)*(this.x[i] - this.neighbours.random().x[i]) +
        a * this.v[i] +
        y * random.normal(0,1);
    }
  }

  updateLocation() {
    for (i from 1 to dimensions) {
      this.x[i] = this.x[i] + this.v[i];
    }
    if (f(this.best) > f(this.x)) {
      this.best = this.x;
    }
  }
}
```

Fig. 1. Particle Swarm Optimization in pseudo-code

of swarm intelligence, are PSO and bird flocking algorithm. Those algorithms have been used in the real world applications in the area of computer graphics and animation [16], detection of outliers [3] or document clustering [6].

PSO was proven to be a suitable algorithm for solving dynamic problems [4] and was applied with success to the DVRP[10, 9] with the combination of 2-Opt algorithm for solving the TSP part of the problem. In this work, we employ a different approach, by using PSO algorithm for both phases and furthermore by applying a different problem encoding in the first phase.

One of our goals was to check the possibility of using a continuous encoding of the DVRP problem, thus enabling the usage of the native (continuous) version of the PSO algorithm. The other goal was to check how good results might be achieved, without using the approximation algorithm and relying only on the PSO in both parts of the problem.

The remainder of the paper is organized as follows: First, in section 2 we briefly summarize the PSO algorithm. In section 3 the DVRP is defined in a formal way. Application of the PSO algorithm and the experimental setup and results are presented in sections 4 and 5, respectively. The last section summarizes the experimental findings and concludes the paper.

2 Particle Swarm Optimization Algorithm

PSO algorithm is an iterative optimization method proposed in 1995 by Kennedy and Eberhart [8] and further studied and developed by many other researchers, e.g., [18], [17], [5]. In short, PSO implements the idea of swarm intelligence to solving hard optimization tasks.

In the PSO algorithm, the optimization is performed by the set of particles which are communicating with each other (see Fig. 2). Each particle has its location and velocity. In every step t a location of particle i, x_t^i is updated based on particle's velocity v_t^i:

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

In our implementation of PSO (based on [1] and [18]) particle's i velocity v_t^i is updated according to the following rule:

$$\begin{aligned} v_{t+1}^i = & u_{U[0;g]}^{(1)}(x_{best}^{neighbours_i} - x_t^i) + u_{U[0;l]}^{(2)}(x_{best}^i - x_t^i) + \\ & u_{U[0;r]}^{(3)}(x_t^i - x_t^{random_i}) + a v_t^i + f y_{N(0;1)}, \end{aligned} \tag{2}$$

where

- $x_{best}^{neighbours_i}$ represents the best location in terms of optimization, found hitherto by the neighbourhood of the ith particle,
- x_{best}^i represents the best location in terms of optimization, found hitherto by the particle i,
- $x_t^{random_i}$ is a location of a randomly chosen particle from the neighbours of the ith particle in iteration t,
- g is a neighbourhood attraction factor,
- l is a local attraction factor,
- r is a repulse factor,
- a is an inertia coefficient,
- a is a fluctuation coefficient,
- $u^{(1)}$, $u^{(2)}$, $u^{(3)}$ are random vectors with uniform distribution from the intervals $[0, g]$, $[0, l]$ and $[0, r]$, respectively,
- y is a random vector with coordinates from standard normal distribution.

Such implementation allows to run algorithm as a classic PSO or as a Repulsive PSO (RPSO). In the classic variant $r = 0$ and $g \neq 0$. In RPSO $r \neq 0$ and $g = 0$ (the attraction of the $x_{best}^{neighbours_i}$ point is changed into repulsion from randomly chosen particle allowing for a greater disperse of a swarm).

3 Problem Definition

In the Dynamic Vehicle Routing Problem one considers a fleet V of n vehicles and a series C of m clients (requests) to be served.

All vehicles $v_i, i = 1, \ldots, n$ are identical and have the same $capacity \in \mathbb{R}$ and the same $speed$[1] $\in \mathbb{R}$. Each vehicle is loaded in one of the k depots[2].

Each depot $d_j, j = 1, \ldots, k$ has a certain $location \in \mathbb{R}^2$ and working hours $(start, end)$, where $0 \leq start < end$.

Each client $c_l, l = 1+k, \ldots, m+k$ has a given $location \in \mathbb{R}^2$, $time \in \mathbb{R}$, which is a period of time when the request becomes available ($\min_{j \in 1,\ldots,k} d_j.start \leq time \leq \max_{j \in 1,\ldots,k} d_j end$), $unloadTime \in \mathbb{R}$, which is the time required to unload the cargo, and the $requestSize \in \mathbb{R}$ - size of the request ($requestSize < capacity$).

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

As previously stated, the goal is to serve the clients (requests), according to their defined constraints, with minimal total cost (travel distance). Because of the assignment of the requests to the vehicles each vehicle will gain the following properties:

- $PlannedClients_t \subset C$, a set of clients planned to be visited (at a given moment t),
- $AssignedClients_t \subset C$, an ordered set of clients (at a given moment t), which have already been visited or will be visited by the vehicle (the order of the set is defined by the order of visits),
- $distance_i \in \mathbb{R}$, a total travel distance of the vehicle,
- $\widehat{distance}_i \in \mathbb{R}$, an estimation of the total travel distance of the vehicle (based on the clients in the $PlannedClients_t$ set but not on the route through them),
- $depot \in \{1, 2, \ldots, k\}$, the index of the depot to which the vehicle will return at the end of working day.

The sets of the $PlannedClients$ and $AssignedClients$ have the following properties:

$$(\forall_{t \in \mathbb{R}})(\forall_{v \in V})(\forall_{c \in C}) \quad c \in v.PlannedClients_t \Leftrightarrow c \notin v.AssignedClients_t \tag{3}$$

$$\begin{aligned} &(\forall_{t \in \mathbb{R}})(\forall_{v, v\prime \in V})(\forall_{c \in v.PlannedClients_t \cup v.AssignedClients_t}) \\ &c \in v\prime.PlannedClients_t \cup v\prime.AssignedClients_t \Rightarrow v = v\prime \end{aligned} \tag{4}$$

The formulae state that each client could be associated with only one vehicle (eq. 4) and only using one type of association (eq. 3).

4 A 2-Phase Particle Swarm Optimization in Dynamic Vehicle Routing Problem

In the rest of the paper we will use the following definitions:
Available vehicles - a set of vehicles to which the requests may still be assigned

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

[2] In all benchmarks used in this paper $k = 1$.

(added) without breaking the time bounds of the problem.
Operating area - an area enclosed by the polygon spanned by the locations of planned and assigned clients.
Cut-off time - a time of the day (within the working hours) after which requests (defined in benchmark sets) are postponed to the next working day (they are available at the next day's start time).

4.1 The Algorithm

The problem will be solved by dividing the working day into pairwise equal time slices and solving partial problems with the use of the best known solution previously found.

The number of time slices n_{ts} equals 25 and cut-off time T_{co} equals 0.5 as proposed and tested by Montemanni et al.[15]. We have stuck to the above-mentioned selection in order to allow a direct comparison of our results with the ones obtained previously for this problem [10, 9, 15, 7]. We also measure the dynamism of the problem (*dod*) used by Khouadjia et al. [10], defined as follows:

$$dod = \frac{\text{Number of requests unknown at the beginning of the working day}}{\text{Number of all requests}} \quad (5)$$

Because of the nature of the problem (which is essentially a combination of two different NPC problems) the optimization algorithm has been divided into two separate phases (stages) in each time slice. In the first phase we assign clients to the vehicles and in the second one optimize a route, separately for each of the vehicles.

The optimization is performed for the client requests known at the moment of the algorithm's start (n_{ts} times a day), i.e. the information about new requests is not updated during the algorithm's run (in a given time step).

The general pseudocode of the algorithm is presented on Fig. 4.1.

4.2 Client Assignment Encoding

The arguments of the fitness function for the first stage are the locations of the centres of the operating areas for the vehicles and the result of the function is a sum of the estimated routes' lengths and optionally an additional penalty for the estimated late return to the depot (i.e. after its closing).

If $x = [x_1, x_2, \ldots, x_{2n}]$ is the argument of the fitness function then the center of the operation area of $vehicle_j$ is defined as follows: $center_j.x = x_{2j-1}$ and $center_j.y = x_{2j}$. The closest (in terms of Euclidean distance between the client location and the vehicle center of the operating area) clients are added to the list of the *planned clients* of the closest **available vehicle**.

The value of the vehicles' routes length estimation equals to the sum of the estimated route length for each of the vehicles $\sum_{v \in V} \widehat{v.distance}$. The vehicle's route length estimation is expressed as a sum of three factors: *assignedRoute*

```
time = depot.start;
while (time < depot.end)
{
  foreach (v in Vehicles)
    v.Available = true;
  while (|UnassignedClients| > 0 and |AvailableVehicles| > 0)
  AssignClientsToAvailableVehicles();
  foreach (v in Vehicles)
  {
    v.OptimizeRoute();
    if (|v.LateClients| > 0)
      v.Available = false;
    v.RemoveLateClientsFromTheRoute(); //mark late clients as unassgined
    v.CommitVehicles();
  }
  time += (depot.end - depot.start) / n_ts;
}
```

Fig. 2. The pseudocode of the algorithm for applying PSO to the DVRP

(length of route through the *assigned clients*), $\widehat{plannedRoute}$ (estimated length of the route through the *planned clients*) and $\widehat{clusterCost}$ (estimated distance from depot to the planned clients multiplied by the doubled number of necessary returns to the depot for reloading the vehicle).

$$v.\widehat{distance} = v.assignedRoute + v.\widehat{plannedRoute} + v.\widehat{clusterCost} \tag{6}$$

$$v.assignedRoute = \sum_{i=2}^{|v.AssignedClients|} \rho\left(v.AssignedClients_{i-1}.location, v.AssignedClients_i.location\right) \tag{7}$$

$$v.\widehat{plannedRoute} = \sum_{i=1}^{|v.PlannedClients|} \rho(v.PlannedClients_i.location, v.center) \tag{8}$$

$$v.\widehat{clusterCost} = \left\lceil \frac{\sum_{c \in v.AssignedClients \cup v.PlannedClients} c.requestSize}{v.capacity} \right\rceil * 2\rho(v.center, v.depot.location) \tag{9}$$

The route through the *planned clients* is estimated by the sum of the distance from the vehicle's operating area center to each of the *planned clients*. And the

distance from the depot to the clients is estimated by the distance from vehicle's operating area center to depot location.

4.3 Managing the Vehicles

Vehicle Route Encoding. The argument of the fitness function for the second stage defines an order of the *planned clients* and the result of the function is the length of the route.

If $x = [x_1, x_2, \ldots, x_{2n}]$ is the argument of the function then the rank of the vehicle's j^{th} *planned client* is equal to the rank of the x_j coordinate in the set $\{x_1, x_2, \ldots, x_{2n}\}$ (if $x_k = x_l \wedge k \neq l$ then the order of the k^{th} and l^{th} clients is defined by the order of l and k).

Committing the Vehicles. After the second phase of the optimization the algorithm returns proposed routes over the *planned clients* set for each of the vehicles. The vehicles with small time reserve[3] are committed to their subsequent *planned clients*.

The set of the clients which are to be moved from the *plannedClients* to the *assigned clients* of the vehicle v is defined as follows:

$$\{client : client \in v.PlannedClients \wedge \\ client.arrivalTime \leq \min_{c \in v.PlannedClients} (c.arrivalTime > nextTimeStep)\}$$

where $arrvialTime$ is the time when the client will be visited by the vehicle v on the planned route.

Keeping Solution Within the Time Bounds. Keeping solution within the time bounds is achieved by adding a simple penalty term $pf(vehicles)$ to the vehicles' routes length estimation function. If despite the pf function, the time bounds are exceeded, the late client requests are assigned to the closest **available vehicle**, thus obtaining the feasible solution.

The penalty function is defined as follows:

$$pf(vehicles) = \sum_{v \in vehicles} \begin{cases} (v.\widehat{time} - v.depot.end)^2, & v.\widehat{time} > v.depot.end \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $v.depot$ is the depot to which the vehicle returns at the end of the working day and $v.\widehat{time}$ is vehicle estimated travel time. $v.\widehat{time}$ is equal to the value of the route length evaluation function for the vehicle[4] plus time required to unload the cargo.

[3] Small time reserve was experimentally set as arriving at the depot later then 3 time steps before the depot closing time

[4] The time can by unified with the distance since the vehicle speed is defined as one distance unit per one time unit.

Table 1. Presentation of the best and $\overline{x}$ (average) results with the S (standard deviation of the sample), ratio to the best known VRP solution accomplished by the proposed method (denoted by 2-phase PSO).

Benchmark[13]	2-phase PSO				
	Best	$\overline{x}$	S	$\frac{\text{Best}}{\text{VRPBest}}$ [2]	dod
c50D.vrp	582.88	675.14	40.62	1.11	0.46
c75D.vrp	912.23	1015.16	51.41	1.09	0.52
c100D.vrp	996.40	1149.48	79.78	1.21	0.59
c100bD.vrp	828.94	850.68	23.70	1.01	0.59
c120D.vrp	1087.04	1212.38	106.80	1.04	0.42
c150D.vrp	1173.94	1336.84	62.72	1.14	0.47
c199D.vrp	1446.93	1578.99	71.24	1.12	0.47
f71D.vrp	315.00	356.75	23.70	1.30	0.59
f134D.vrp	12813.14	13491.60	319.41	1.10	0.57
tai75aD.vrp	1871.06	2142.07	162.75	1.16	0.56
tai75bD.vrp	1460.95	1568.21	69.80	1.09	0.56
tai75cD.vrp	1500.23	1811.08	152.80	1.16	0.56
tai75dD.vrp	1462.82	1586.28	95.95	1.07	0.37
tai100aD.vrp	2317.76	2707.61	202.75	1.13	0.56
tai100bD.vrp	2187.86	2510.60	167.79	1.13	0.56
tai100cD.vrp	1564.25	1672.33	68.78	1.11	0.46
tai100dD.vrp	1859.70	2220.01	147.32	1.18	0.49
tai150aD.vrp	3638.75	4151.31	314.32	1.19	0.46
tai150bD.vrp	3107.95	3302.94	126.36	1.14	0.49
tai150cD.vrp	2781.02	2952.88	68.80	1.18	0.49
tai150dD.vrp	3048.24	3478.49	225.68	1.15	0.56
Sum/Avarage	46957.09	51770.84	2582.48	1.13	0.51

Passing Down Historic Knowledge. As the center of the search space is biased in the search for the global optimum [12, 19], the center of the search space for the particular time slice is chosen based on the best known solution from the previous time slice while missing arguments (parameters defining the order of requests that were not present before) are generated at random with the underlying condition that at least one particle's initial location is equal to the best solution previously found.

5 Tests and Results

Benchmark Sets. The algorithm was tested on the same benchmark sets as the ones used in [10, 9, 15, 7]. The best achieved value, the mean value, the standard deviation, the relation to the best known static solution and the value of *dod* for each of the tested problems are presented in Table 1. Table 2 shows the best achieved values of the algorithm compared to other PSO approaches [10, 9] for solving the DVRP.

Table 2. Comparison of the best and average results accomplished by various approaches and the best known solutions for the static version of considered benchmark problems. The best results in each case are bolded. Our method is presented as 2-phase PSO.

Benchmark[13]	Algorithm						
	2-phase PSO		MAPSO[9]		DAPSO[10]		VRP[2]
	Best	Avarage	Best	Avarage	Best	Avarage	
c50	582.88	675.14	**571.34**	**610.67**	575.89	632.38	524.61
c75	**912.23**	1015.16	931.59	**965.53**	970.45	1031.76	835.26
c100	996.40	1149.48	**953.79**	**973.01**	988.27	1051.50	826.14
c100b	**828.94**	**850.68**	866.42	882.39	924.32	964.47	819.56
c120	**1087.04**	**1212.38**	1223.49	1295.79	1276.88	1457.22	1042.11
c150	**1173.94**	**1336.84**	1300.43	1357.71	1371.08	1470.95	1028.42
c199	**1446.93**	**1578.99**	1595.97	1646.37	1640.40	1818.55	1291.45
f71	315.00	356.75	287.51	**296.76**	**279.52**	312.35	241.97
f134	**12813.14**	**13491.60**	15150.50	16193.00	15875.00	16645.89	11629.60
tai75a	1871.06	2142.07	**1794.38**	**1849.37**	1816.07	1935.28	1618.36
tai75b	1460.95	1568.21	**1396.42**	**1426.67**	1447.39	1484.73	1344.64
tai75c	1500.23	1811.08	1483.10	**1518.65**	**1481.35**	1664.40	1291.01
tai75d	1462.82	1586.28	**1391.99**	**1413.83**	1414.28	1493.47	1365.42
tai100a	2317.76	2707.61	**2178.86**	**2214.61**	2249.84	2370.58	2047.90
tai100b	2187.86	2510.60	**2140.57**	**2218.58**	2238.42	2385.54	1940.61
tai100c	1564.25	1672.33	**1490.40**	**1550.63**	1532.56	1627.32	1407.44
tai100d	1859.70	2220.01	**1838.75**	**1928.69**	1955.06	2123.90	1581.25
tai150a	3638.75	4151.31	**3273.24**	**3389.97**	3400.33	3612.79	3055.23
tai150b	3107.95	3302.94	**2861.91**	**2956.84**	3013.99	3232.11	2727.99
tai150c	2781.02	2952.88	**2512.01**	**2671.35**	2714.34	2875.93	2362.79
tai150d	3048.24	3478.49	**2861.46**	**2989.24**	3025.43	3347.60	2655.67
Sum	**46957.09**	51770.84	48104.13	**50349.66**	50190.87	53538.72	41637.43

Tests Configuration. For all tests the system was run with the following parameters, suggested in [5, 18]: $g = 0.6$, $l = 2.2$, $r = 0.0$, $a = 0.63$, $f = 0.0$ in eq. (2). A random star topology with probability $P(X \text{ is a neighbour of } Y) = 0.7$ was applied[5].

The algorithm was run 150 times for each of the benchmark problems. In each case, the number of particles was equal to 40, the number of iterations for each time slice was restricted to 500 (in each of the two phases of the algorithm). Thus the total number of function evaluations for the first phase was equal to 500 000 and the number of function evaluations for the second phase was about 4 000 000 (depending on the number of vehicle routes).

Choice of the Parameters. Parameter $P(X \text{ is a neighbour of } Y) = 0.7$ was empirically chosen from the set of $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ based on a small

[5] Note, that the fact that X is a neighbour of Y does not mean that Y is a neighbour of X.

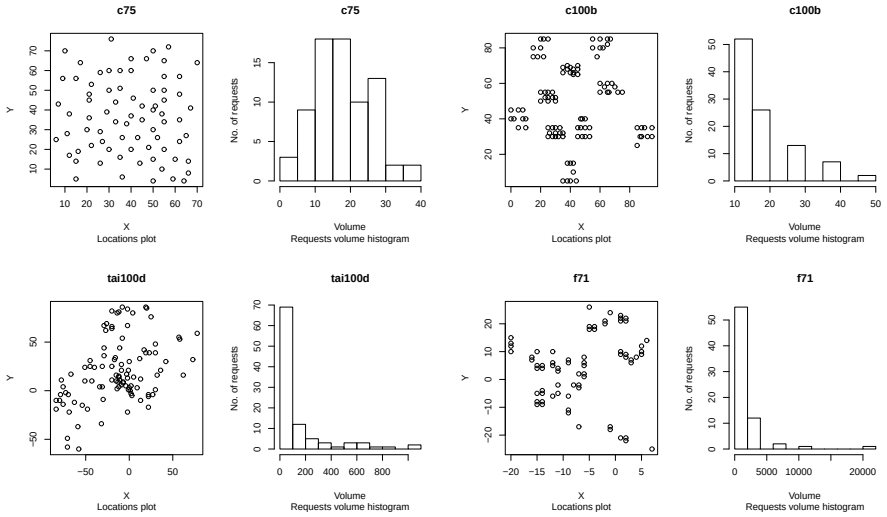

Fig. 3. Benchmark problems examples with clients' requests having spatially uniform location and symmetric volume distributions (c75), spatially clustered location and skewed volume distributions (c100b), spatially semiclustered location and largely skewed volume distributions (tai100d), and spatially clustered location and largely skewed volume distributions (f71)

number of runs on benchmark problem's instances. The other parameters were left at their suggested values (from the literature).

6 Discussion and Conclusions

The DVRP discussed in this paper is an important problem not only because of its real world applications but also a possibility to study continuous optimization algorithms in solving dynamic combinatorial tasks.

The proposed algorithm achieved better results than other PSO approaches in 6 out of 21 test problems. On Christofides benchmark set, 2-phase PSO best solution was averaged out at 0.96 times the length of the MAPSO and DAPSO best results. On Fisher and Taillard benchmark sets it was 0.99 and 1.06, respectively. The overall result was on average 1.02 times longer in comparison with the best results from 2-phase PSO and other approaches and 1.09 times longer in terms of the average results.

Closer analysis of the clients' requests locations and volume distributions reveals that the proposed algorithm performs better for more spatially clustered benchmarks (e.g. c100b) and for more symmetric volume distributions (e.g. c75).

The algorithm's worst performance on the f71 benchmark set, may be related to the configuration of the largest and the second-largest request in this instance of a DVRP. Those two requests are in a close distance to each other and the joint volume of those requests slightly exceeds the capacity of a vehicle, making the BPP part of the task hard to solve efficiently. Moreover the largest request is

a strong outlier in terms of the request volume. It is more then twice the size of the second-largest request in this particular set. In all other test sets this ratio never exceeds 1.37. The examples of the clients' requests location and volume distributions are presented in Fig. 3.

Overall, the results suggest that the DVRP may be efficiently solved by the PSO even without its hybridization with 2-Opt algorithm.

Further research of the proposed algorithm will include the use of alternative encodings of the clients' assignments, further analysis of the impact of locations and requests distribution on the algorithm's performance, and application of a multi-swarm approach.

Acknowledgements. Study was supported by research fellowship within "Information technologies: Research and their interdisciplinary applications" agreement number POKL.04.01.01-00-051/10-00.

The analysis of the requests distribution within the tests sets has been done with the usage of R[14].

References

1. Standard PSO 2011 (2012), http://www.particleswarm.info/
2. Vrp web (2012), http://neo.lcc.uma.es/radi-aeb/WebVRP/
3. Bellaachia, A., Bari, A.: A flocking based data mining algorithm for detecting outliers in cancer gene expression microarray data. In: Information Retrieval Knowledge Management (CAMP), 2012 International Conference on. pp. 305 –311 (march 2012)
4. Blackwell, T.: Particle swarm optimization in dynamic environments. Evolutionary Computation in Dynamic and Uncertain Environments 51, 29 – 49 (2007)
5. Cristian, I., Trelea: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317 – 325 (2003)
6. Cui, X., Potok, T., Palathingal, P.: Document clustering using particle swarm optimization. In: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. pp. 185 – 191 (june 2005)
7. Hanshar, F.T., Ombuki-Berman, B.M.: Dynamic vehicle routing using genetic algorithms. Applied Intelligence 27(1), 89–99 (Aug 2007), http://dx.doi.org/10.1007/s10489-006-0033-z
8. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks. IV pp. 1942–1948 (1995)
9. Khouadjia, M., Alba, E., Jourdan, L., Talbi, E.G.: Multi-swarm optimization for dynamic combinatorial problems: A case study on dynamic vehicle routing problem. In: Swarm Intelligence, Lecture Notes in Computer Science, vol. 6234, pp. 227–238. Springer, Berlin / Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15461-4_20
10. Khouadjia, M.R., Sarasola, B., Alba, E., Jourdan, L., Talbi, E.G.: A comparative study between dynamic adapted pso and vns for the vehicle routing problem with dynamic requests. Applied Soft Computing 12(4), 1426 – 1439 (2012), http://www.sciencedirect.com/science/article/pii/S1568494611004339
11. Kilby, P., Prosser, P., Shaw, P.: Dynamic vrps: A study of scenarios (1998), http://www.cs.strath.ac.uk/~apes/apereports.html

12. Monson, C.K., Seppi, K.D.: Exposing origin-seeking bias in pso. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. pp. 241–248. GECCO '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1068009.1068045
13. Pankratz, G., Krypczyk, V.: Benchmark data sets for dynamic vehicle routing problems (2009), http://www.fernuni-hagen.de/WINF/inhalte/benchmark_data.htm
14. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2012), http://www.R-project.org/, ISBN 3-900051-07-0
15. R. Montemanni, L. Gambardella, A.R.A.D.: A new algorithm for a dynamic vehicle routing problem based on ant colony system. Journal of Combinatorial Optimization
16. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. 21(4), 25–34 (Aug 1987), http://doi.acm.org/10.1145/37402.37406
17. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation p. 69 73 (1998)
18. Shi, Y., Eberhart, R.: Parameter selection in particle swarm optimization. Proceedings of Evolutionary Programming VII (EP98) p. 591 600 (1998)
19. Spears, W. M., G.D.T..S.D.F.: Biases in particle swarm optimization. International Journal of Swarm Intelligence Research 1(2), 34–57 (2010)