Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 32 (2007) 508–517
Expert Systems with Applications
www.elsevier.com/locate/eswa

A multi-parametric evolution strategies algorithm for vehicle routing problems

David Mester [a,*], Olli Bräysy [b], Wout Dullaert [c]

[a] Institute of Evolution, Mathematical and Population Genetics Laboratory, University of Haifa, 31905 Haifa, Israel
[b] Agora Innoroad Laboratory, Agora Center, P.O. Box 35, FI-40014 University of Jyväskylä, Finland
[c] Institute of Transport and Maritime Management Antwerp, University of Antwerp, Keizerstraat 64, B-2000 Antwerp, Belgium

Abstract

Vehicle routing problems are at the heart of most decision support systems for real-life distribution problems. In vehicle routing problem a set of routes must be determined at lowest total cost for a number of resources (i.e. fleet of vehicles) located at one or several points (e.g. depots, warehouses) in order to efficiently service a number of demand or supply points. In this paper an efficient evolution strategies algorithm is developed for both capacitated vehicle routing problem and for vehicle routing problem with time window constraints. The algorithm is based on a new multi-parametric mutation procedure that is applied within the $1 + 1$ evolution strategies algorithm. Computational testing on six real-life problems and 195 benchmark problems demonstrate that the suggested algorithm is efficient and highly competitive, improving or matching the current best-known solution in 42% of the test cases.

Keywords: Distribution management; Vehicle routing problem; Heuristics; Evolution strategies

1. Introduction

The vehicle routing problem (VRP) is one of the most significant problems in distribution management. Its objective is to find the optimal routes for distributing various shipments, such as goods, mail and raw materials. The basic VRP consists of a number of geographically scattered customers, each requiring a specified weight (or volume) of goods to be delivered (or picked up). A fleet of identical vehicles dispatched from a single depot is used to deliver the goods required and once the delivery routes have been completed, the vehicles must return to the depot. Each vehicle can carry a limited weight and only one vehicle is allowed to visit each customer. It is assumed that all problem parameters, such as customer demands and travel times between customers are known with certainty. Solving the problem consists of finding a set of delivery routes which satisfy the above requirements at minimal total cost. In the literature the above described problem is called capacitated VRP (CVRP). In CVRP the total cost equals the total distance or travel time. In addition to CVRP we also consider in this paper another central distribution management problem, the vehicle routing problem with time windows (VRPTW). Compared to CVRP, in VRPTW each customer can be serviced only within a specified time interval or time window, and a service time is assigned to each customer, corresponding to the loading or unloading time. The VRPTW involves typically a hierarchical objective function where the primary objective is to minimize the number of routes whereas the minimization of total distance or travel time is the secondary objective.

Both the CVRP and the VRPTW are at the heart of many real-life distribution problems. Some of the most important applications of CVRP and VRPTW are minimization of distribution costs in supply chains, routing of automated guiding vehicles, rolling batch planning, bank and postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, security patrol services and JIT (just in time) manufacturing.

* Corresponding author. Tel.: +972 04 8288 678; fax: +972 04 8246 554.
E-mail address: dmester@research.haifa.ac.il (D. Mester).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.12.014

The significant economic importance and mathematical complexity renders both problems of great interest in combinatorial optimization. As the VRP is a very complex NP-hard problem, solving the real-life VRPs to optimality is often not possible within the limited computing time available in practical situations. Therefore, most of the research has focused on heuristic and metaheuristic solution methods designed to produce high quality solutions in a limited time. For more details, we refer to the extensive surveys by Bräysy, Hasle, and Dullaert (2004a, 2004b, 2005a, 2005b), Cordeau, Gendreau, Laporte, Potvin, and Semet (2002, 2005), Gendreau, Laporte, and Potvin (2001), and Laporte and Semet (2001).

Given the practical importance of VRPs, it is crucial that new solution algorithms are developed to solve VRPs as efficiently as possible. The main contribution of this paper is the development of new multi-parametric evolution strategies algorithm for CVRP and VRPTW and comprehensive computational study to demonstrate the efficiency of the suggested method on six real-life problems as well as on 195 benchmark problems from the corresponding literature.

The remainder of this paper is outlined as follows. In Section 2 we summarize the main ideas of the evolution strategies metaheuristic and the improvement heuristics applied within our algorithm. Then we proceed to the description of the main structure of the solution algorithm, the initial solution procedure and finally the multi-parametric evolution strategies that are used to improve the initial solution. In Section 3 we present the results of an extensive computational study on real-life and theoretical benchmarks and in Section 4 conclusions are made.

2. The problem solving methodology

The proposed heuristic solution method consists of two phases. In the first phase, a new hybrid cheapest insertion heuristic is used to construct a feasible initial solution. After the construction, a set of standard improvement heuristics are applied within a composite to improve the quality of the initial solution. After generating the initial solution, an attempt is made to improve the solution in the second phase with a new multi-parametric $(1 + 1)$-evolution strategies metaheuristic. The second phase is continued until no further improvement can be found and then the search is stopped. Next we will briefly describe the main features of the evolution strategies metaheuristics and the improvement heuristics used before proceeding to the actual description of the global heuristic.

2.1. Evolution strategies

Evolution strategies (ES) belong to the class of evolutionary solution algorithms. As the name implies, evolutionary algorithms mimic features of natural evolution, such as adapting to the environment and passing on genetic information to future generations. In general, an evolutionary algorithm (EA) is characterized by maintaining a set of solution candidates that undergoes a selection process, and is manipulated by genetic operators. Natural evolution is simulated by an iterative computation process. To start a population of candidate solutions to one specific problem is initialized. This is often accomplished with a tailored heuristic or by randomly sampling from the solution space. Then a loop consisting of evaluation of solution candidates and selection of solutions based on the evaluation and manipulation of the selected solutions with genetic operators is executed a certain number of times. Each loop iteration is called a generation, and the search is typically stopped after a given number of generations or after a convergence to a homogeneous population.

ES were originally developed by Rechenberg (1973) and Schwefel (1977) to optimize real-value engineering design problems. ES are typically characterized with a notation (λ, μ) where μ is the population size and λ is the number of new solutions (offspring) created at each generation. We applied here a $(1 + 1)$ ES, meaning that the population consists of only one solution, and at each generation only one new solution is generated. To be more precise, at each generation small modifications are applied to the current solution in the population using a genetic operator called multi-parametric mutator. As a result, a set of modified solutions is obtained. Next, the generated new solutions are compared against the current "parent" solution and should any of the new solutions prove better than the parent, this replaces the parent in the population. Otherwise, the parent solution is kept. The above selection process is deterministic. The multi-parametric mutator works directly on the actual solutions and the size and type of attempted modifications to the parent are varied dynamically during the search based on the parameter setting of the mutator. The search is stopped if no further improvements can be found by the ES. For vehicle routing problems, ES have been previously used by Gehring and Homberger (1999, 2001), Homberger and Gehring (1999, 2005) and Mester and Bräysy (2005).

2.2. Improvement heuristics

We recall here the main features of the improvement heuristics applied within the initial construction heuristic and the multi-parametric mutator. Four of the used five heuristic procedures (forward Or-exchange, backward Or-exchange (Or, 1976), 1-interchange (Osman, 1993) and 2-opt* (Potvin & Rousseau, 1995) are used solely for inter-route improvements, in other words they modify two routes simultaneously whereas 2-opt (Flood, 1956) works on one single route at a time.

The basic idea of forward and backward Or-exchange procedures is to reinsert one single customer at a time in an alternate position in the solution vector. Here the solution vector consists of an ordered list of all customer indexes (integers). The solution vector is divided in r subsequent sets (routes) that are in a determined order based on

the previously created solution. Forward Or-exchange considers reinsertions only to positions in alternate routes located after the present route of the customer in the current solution vector. Correspondingly, backward Or-exchange attempts reinsertions only to routes located before the origin route in the solution vector. The 1-interchange swaps simultaneously the position of two customers in two different routes and 2-opt* swaps the end (or start) portions of two routes by replacing one edge in both routes with a new one. In other words, it combines two routes so that the last customers of a given route are introduced after the first customers of another route in the order they were in their route of origin. The 2-opt heuristic tries to improve a single route by replacing two of its edges by two other edges. The improvement heuristics are applied here with the best-accept strategy, i.e. all possible moves in the current neighborhood are evaluated and the best improving move is selected.

The above described improvement heuristics are applied together in a set called composite. The heuristics in the composite set are repeated in a loop until no further improvement can be found such that after each successful move the heuristic is changed. In the remainder of the paper this composite procedure is called composite local search. Here one must note that for the VRPTW the 2-opt is not applied as it often causes violation of the time window constraints.

2.3. Initial solution procedure

The initial solution for the second phase is created with a new cheapest insertion heuristic. The heuristic begins with an initial solution in which each customer is supplied individually by a separate route, i.e. the number of routes equals the number of customers. Reinsertions of single customers to alternative positions in the solution vector are then attempted in a loop starting from the beginning of the solution vector. For a customer k currently serviced between customers i and j, the heuristic considers only positions between the adjacent customers l and m, such that $k_p < l_p < m_p \leqslant n$ where k_p, l_p and m_p refer to the positions of customers k, l and m in the current solution vector with n customers. The reinsertions are evaluated using a modified insertion criterion of Osman, 1993.

$$(c_{ik} + c_{kj} - c_{lm}) - \alpha \cdot (c_{lk} + c_{km} - c_{ij}), \tag{1}$$

where c_{ij} refers to the costs of the associated arcs and α is a parameter whose values depend on the problem size. For the CVRP and 200-customer VRPTW instances all values in range 0.2–1.4 are tried in increments of 0.2 units, resulting in seven initial solutions. For the 400-customer VRPTW benchmarks five initial solutions are created by varying α in range 0.6–1.4. The reinsertion with the maximum positive value is executed and reinsertions with negative value of the criterion (1) are disregarded. The search is stopped if the algorithm cannot find a route wherein every customer can be inserted in alternative routes, i.e. if one cannot reduce the number of routes. During the construction the composite local search is applied with probability 0.3 until no further improvement can be found. In the end the best initial solution found with different values of α is selected as the starting solution for the second phase.

2.4. The multi-parametric (1 + 1) evolution strategies

In this section we described the multi-parametric ES (MP-ES) applied in the second phase to improve the initial solution. The MP-ES is based on the well-known ruin and recreate search principle. The basic idea is to first remove a set of customers from the solution and then reinsert the removed customers at optimal cost. The removal and reinsertion operations are guided by three parametric rules. We call the procedure applied here remove-insert mutation mechanism and it is detailed in the next subsection. In addition, the mutation takes place in a special parameterized dynamic neighborhood, called adaptive variable neighborhood (AVN). The AVN is described in more detail in Section 2.4.2.

2.4.1. Remove-insert mutation mechanism

The basic idea of the remove-insert mutation mechanism is to remove a selected set of customers from the current solution and then reinsert the removed customers. Removal and reinsertion procedures are considered always only within the AVN, as described Section 2.4.2. Three strategies (RS) are applied to select the customers to be removed:

Strategy 1: Select $\beta = (0.2 + 0.5a)n_{\mathrm{AVN}}$ customers randomly from the current AVN (a is a random value uniformly distributed between 0 and 1).

Strategy 2: Select at most $\beta = (0.2 + 0.5a)n_{\mathrm{AVN}}$ customers from the current AVN within a ring created by two circles with random radiuses, centered at the depot.

Strategy 3: Select all customers n_{AVN} in the current AVN.

After the removal, the selected customers are reinserted using the cheapest insertion heuristic described in Section 2.3. During the reconstruction the composite local search is applied periodically with probability 0.3. The composite local search is also applied after all customers have been inserted back into the solution. Each entity of removing and reinserting a set of customers and improving the obtained solution with the composite local search is called an iteration. The remove-insert mutation is always repeated for 14 iterations at a time. During the first seven iterations only the first two removal strategies are applied with 50% probability. In the last seven iterations the algorithm uses only the third removal strategy. During both sets of seven iterations, the value of parameter α is varied in the ranges 0.2–1.4 and 0.6–1.4 for CVRP and 200-customer VRPTW problems and for 400-customer VRPTW instances, respectively. The value α is increased in increments of 0.2 units. Once all 14 neighboring offspring solutions have been created, the generated solutions are

evaluated and compared against the parent solution. Should a better solution be found, the parent solution is replaced. The search is continued until no further improvements have been found for a given number of iterations.

2.4.2. The adaptive variable neighborhood

In this section we describe the rationale and principles for forming the AVNs. The main rationale for decomposing the problem in AVNs is to speed up the search by limiting the search space and to focus the search on the most promising parts of the search space. All search efforts in the second phase are done within one AVN at a time.

Two interrelated rules are used to create the AVNs. The first rule divides the problem in smaller geographic parts G_i using rectangles and it is applied only in problems having more than 100 customers and more than four routes. Each partitioned geographical region is then considered as a separate AVN and the AVNs are considered in a given order, as shown in Fig. 1.

As can be seen from the leftmost part of the figure, the problem is first divided in four rectangular parts (G_1-G_4) each containing the routes within its geographical area. The four rectangles are formed simply by using the x- and y-coordinates of the depot to draw two orthogonal lines in the direction of x- and y-axels. After repeating the remove-reinsert mechanism in each of the four parts until no further improvements can be found, regions G_1 and G_2 and G_3 and G_4 are merged to G_5 and G_6, as shown in the figure, and the search is restarted using the larger neighborhoods. Correspondingly in the third step regions G_1 and G_3 and G_2 and G_4 are merged to G_7 and G_8 and in the last step the whole problem G_9 is considered without geographical division, as illustrated in Fig. 1. The rationale is that at the beginning the search is more time-consuming as there is more potential for improvements, so one can gain significant acceleration as the decomposition progresses. Later in the search when the obtained solution is already near-optimal, larger neighborhoods can be explored at a time.

The above described geographical division is combined with another mechanism, called dichotomous route combinations (DRC). The basic idea of DRC is to randomly form up combinations of geographically neighboring routes within the above described rectangles and consider improvements only within the formed route combinations. To be more precise, the routes within each rectangle are ordered randomly at the beginning of each iteration. Then a given number of consecutive routes, according to the created order, are collected to a combination. The number of consecutive routes combined together is limited to 2^{δ} where δ is a parameter of dichotomy. All values of δ are considered in the range 0-INT($\log_2(r) - 1$) where r is the number of routes in the considered rectangle. Together the DRC and the geographical division form the AVN. In case an improved solution is found, the search is repeated using the same AVN until no further improvements can be found.

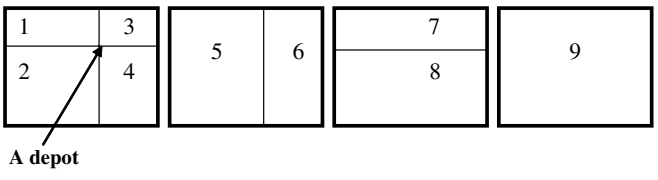

Fig. 1. The forming of AVNs with rectangles.

3. Computational results

3.1. Problem data and the experimental setting

The proposed algorithm was been implemented in Visual Basic 6.0 and the algorithm was executed on a Pentium III 800 MHz computer. The computational tests were carried out with seven standard CVRP benchmarks from Christofides et al. (1979) (without maximum route length constraint) and 12 CVRP benchmark instances from Rochat and Taillard (1995). In addition, the 56 standard VRPTW benchmarks from Solomon (1987) and 120 extended VRPTW instances from Gehring and Homberger (1999) were examined. Finally, to assess the performance of the suggested solution method on real-life cases, it was tested on six real-life problems, taken from Fisher (1994), Russell (1995) and Taillard (1993). The problems vary in the number of customers, fleet size, vehicle capacity, spatial and temporal distribution of customers, time window density and width and customer service times. Both travel times and distances separating two customers correspond to their relative Euclidean distance.

The experimental tests consist of only one simulation run for each problem and the parameter values described in the previous section were determined experimentally over a few intuitively selected combinations. The limited parameter testing is justified by the fact that the sensitivity of the results with respect to changes in the parameter values was found to be generally quite small.

The developed system includes a graphical user-interface (see Fig. 2) that can be used to adjust the working of the algorithm in detail, and to present the obtained solution values and the actual solutions graphically.

3.2. Results for the capacitated vehicle routing benchmark problems

The results for the CVRP test instances are presented in Table 1. In the Table the problems set out by Christofides et al. (1979) are listed with notations C1–C5 and C11–C12. In the first five problems the customers are randomly distributed, whereas problems C11 and C12 are structured in the sense that customers appear in clusters. In the T75, T100 and T150 problem sets drawn from Rochat and Taillard (1995) the customers are spread in several clusters with variable size and compactness and the quantities ordered by the customers are exponentially distributed. The

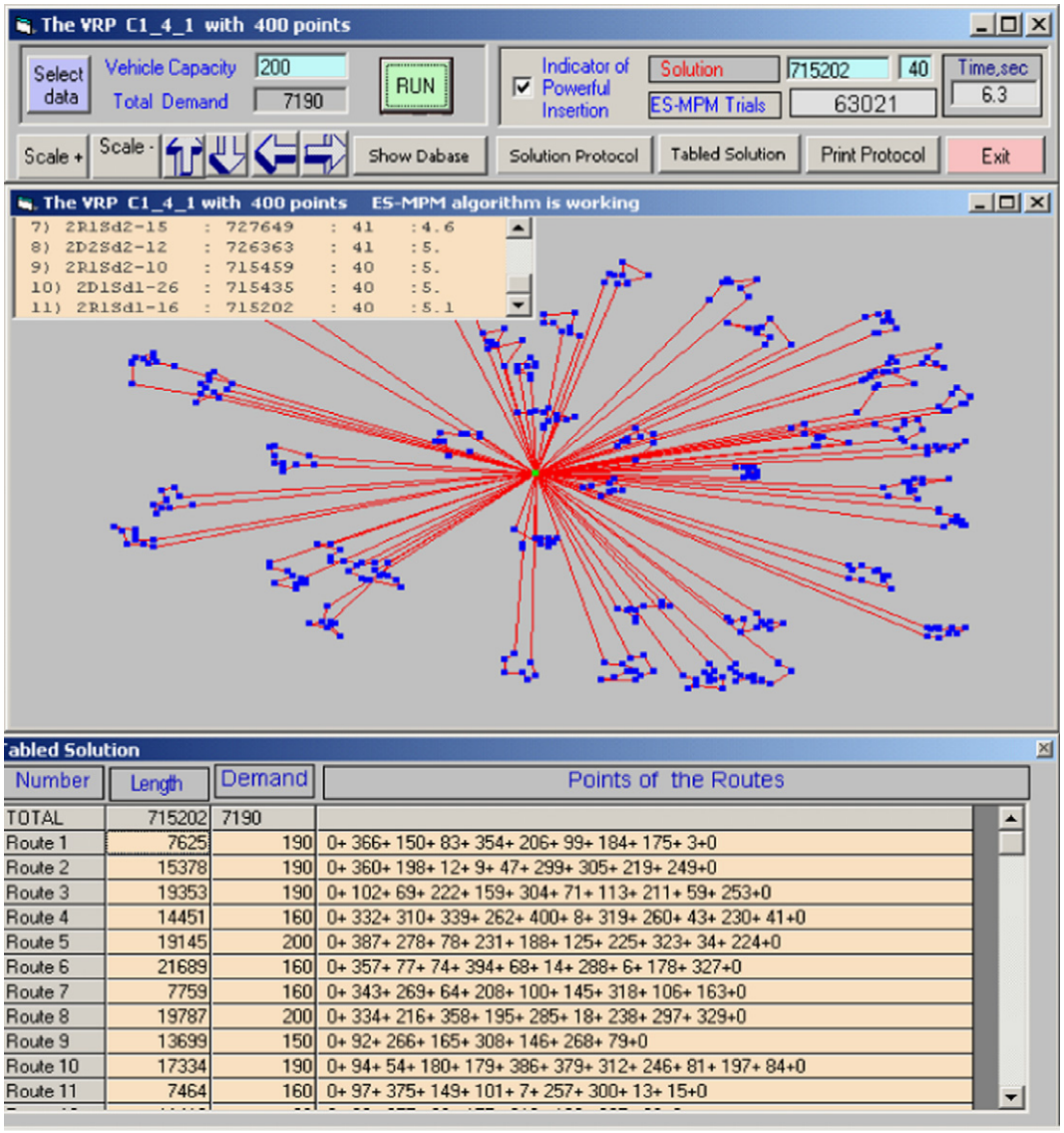

Fig. 2. Graphical user-interface of the developed solution method.

number of customers in each problem is described in the second column of the table. The best-published solutions are taken from Mester and Bräysy (2005). The values in Table 1 refer to the total distance over all routes in a solution that is often considered as the only objective in CVRP. Our results over a single test run are shown in MP-ES column, followed by the difference to the best-known solutions in percents. Finally, the rightmost column displays the CPU time in seconds.

As shown in Table 1, our algorithm exactly repeated the best-known solution for 14 out of the 19 problems. On average the results of MP-ES are only 0.11% above the best-known solutions. The average computing time of MP-ES is 16.1 minutes.

3.3. Results for the VRPTW benchmark problems

The results of MP-ES for the 176 VRPTW benchmarks taken from Solomon (1987) and Gehring and Homberger (1999) are presented in Tables 2–4. Solomon's problems have a hundred customers, served from a central depot, with a fleet of identical vehicles with a given weight capacity. The customers must be served within given time window limits. The problems are divided in six classes. The C1 and C2 classes have customers located in clusters and in the R1 and R2 classes the customers are at random positions. The RC1 and RC2 classes contain a mix of both random and clustered customers. In terms of time window density (the percentage of customers with delivery time windows), the problems have 25%, 50%, 75% and 100% time windows. The C1, R1 and RC1 problems have a short scheduling horizon and require 9–19 vehicles. Short horizon problems have vehicles that have small capacities, and short route times, and therefore cannot service many customers in a single route. Classes C2, R2 and RC2 are more representative of "long-haul" delivery with longer scheduling horizons and fewer (2–4) vehicles. Gehring and Homberger (1999) introduced an extended set of Solo-

Table 1
Results of MP-ES on standard CVRP instances

Problem	Problem size	Best published	MP-ES	% Above best-known	CPU/s
C1	50	524.61	524.61	0	0.5
C2	75	835.26	835.26	0	12.0
C3	100	826.14	826.14	0	32.0
C4	150	1028.42	1028.42	0	6830.0
C5	199	1291.29	1294.30	0.233	1300.0
C11	120	1042.11	1042.17	0.006	40.0
C12	100	819.56	819.56	0	1.7
T75a	75	1618.36	1618.36	0	3.0
T75b	75	1344.64	1355.15	0.782	44.3
T75c	75	1291.01	1291.01	0	5.0
T75d	75	1365.42	1365.42	0	1.7
T100a	100	2041.34	2041.34	0	132.0
T100b	100	1939.90	1939.90	0	16.0
T100c	100	1406.20	1406.20	0	133.0
T100d	100	1581.25	1581.25	0	1366.0
T150a	150	3055.23	3055.23	0	1850.0
T150b	150	2727.67	2727.77	0.004	2033.0
T150c	150	2341.84	2361.56	0.842	2720.0
T150d	150	2645.40	2645.40	0	1850.0
Average		1564.51	1566.27	0.11	966.85

Table 2
Comparison of results of different heuristic algorithms applied to Solomon's problems

Reference	R1	R2	C1	C2	RC1	RC2	Cumulative	CPU
Homberger and Gehring (1999)	11.92	2.73	10.00	3.00	11.63	3.25	406	Pentium 200
	1228.06	969.95	828.38	589.86	1392.57	1144.43	57 876	10 runs, 13 min
Gehring and Homberger (2001)	12.00	2.73	10.00	3.00	11.50	3.25	406	Pentium 400
	1217.57	961.29	828.63	590.33	1395.13	1139.37	57 641	5 runs, 413.5 min
Bräysy (2003)	11.92	2.73	10.00	3.00	11.50	3.25	405	Pentium 200
	1222.12	975.12	828.38	589.86	1389.58	1128.38	57 710	1 run, 82.5 min
Bräysy et al. (2004a, 2004b)	12.00	2.73	10.00	3.00	11.50	3.25	406	AMD 700
	1220.20	970.38	828.38	589.86	1398.76	1139.37	57 796	3 runs, 2.6 min
Bent and Van Hentenryck (2004)	12.17	2.73	10.00	3.00	11.63	3.25	409	Sun Ultra 10
	1203.84	980.31	828.38	589.86	1379.03	1158.91	57 707	5 runs, 30 min
Homberger and Gehring (2005)	12.08	2.82	10.00	3.00	11.50	3.25	408	Pentium 400
	1211.67	950.72	828.45	589.96	1395.93	1135.09	57 422	3 runs, 1.6 min
Le Bouthillier and Crainic (2005)	12.08	2.73	10.00	3.00	11.50	3.25	407	Pentium 933
	1209.19	963.62	828.38	589.86	1389.22	1143.70	57 412	1 run, 512 min
Ibaraki et al. (2005)	11.92	2.73	10.00	3.00	11.63	3.25	406	Pentium 1000
	1220.02	961.64	828.38	589.86	1378.72	1132.17	57 480	1 run, 33 min
Pisinger and Röpke (2005)	11.92	2.73	10.00	3.00	11.50	3.25	405	Pentium 3000
	1212.39	957.72	828.38	589.86	1385.78	1123.49	57 332	10 runs, 2.4 min
MP-ES	12.00	2.73	10.00	3.00	11.50	3.25	406	Pentium 800
	1208.18	954.09	828.38	589.86	1387.12	1119.70	56 812	1 run, 43.8 min

mon's problems with up to 1000 customers. In this paper we consider the 200- and 400-customer problem sets. Both sets have 60 problems, divided in R1, R2, C1, C2, RC1 and RC2 subclasses as in Solomon (1987). The first column of Tables 2–4 gives the reference of the authors consulted. Columns R1, R2, C1, C2, RC1 and RC2 present the average number of vehicles and average total distance with respect to the six problem groups taken from Solomon (1987). The cumulative column indicates the cumulative number of vehicles and cumulative total distance over all problems in the sets. Finally, the rightmost column describes the computer, number of independent runs and the CPU time in minutes, as reported by the authors. All methods in the tables consider the number of vehicles as the primary objective and the total distance as the secondary objective.

Table 2 shows that the MP-ES generates competitive results for the 100 customer problem instances. Compared to previous best approaches with the same cumulative number of vehicles MP-ES reduces total distance by more than 1.16%. Moreover, MP-ES has generated nine new best solutions which are listed in Table 5. Only Bräysy (2003)

Table 3
The results for the 200-customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	Cumulative	CPU
Gehring and Homberger (1999)	18.20	4.00	18.90	6.00	18.00	4.30	694	Pentium 200
	3705	3055	2782	1846	3555	2675	176 180	1 run, 410 min
Gehring and Homberger (2001)	18.20	4.00	18.90	6.00	18.10	4.40	696	Pentium 400
	3855.03	3032.49	2842.08	1856.99	3674.91	2671.34	179 328	3 runs, 42.1 min
Li and Lim (2003)	18.30	4.10	19.10	6.00	18.30	4.90	707	Pentium 545
	3736.20	3023.00	2728.60	1854.90	3385.80	2518.70	172 472	3 runs, 182.1 min
Bräysy et al. (2004a, 2004b)	18.20	4.00	18.90	6.00	18.00	4.40	695	AMD 700
	3718.30	3014.28	2749.83	1842.65	3329.62	2585.89	172 406	3 runs, 2.4 min
Bent and Van Hentenryck (2004)	18.20	4.10	18.90	6.00	18.00	4.50	697	Sun Ultra 10
	3677.96	3023.62	2726.63	1860.17	3279.99	2603.08	171 715	N/A, N/A
Homberger and Gehring (2005)	18.20	4.10	19.00	6.00	18.10	4.50	699	Pentium 400
	3890.06	3059.78	2836.66	1898.44	3734.32	2640.94	180 602	3 runs, 1.6 min
Le Bouthillier and Crainic (2005)	18.20	4.00	18.90	6.00	18.00	4.30	694	Pentium 933
	3676.95	2986.01	2743.66	1836.10	3449.71	2613.75	173 061	1 run, 510 min
Mester and Bräysy (2005)	18.20	4.00	18.80	6.00	18.00	4.40	694	Pentium 2000
	3618.68	2942.92	2717.21	1833.57	3221.34	2519.79	168 573	1 run, 8 min
Pisinger and Röpke (2005)	18.20	4.00	18.90	6.00	18.00	4.30	694	Pentium 3000
	3631.23	2949.37	2721.52	1832.95	3212.28	2556.87	169 042	10 runs, 7.7 min
MP-ES	18.20	4.00	18.90	6.00	18.00	4.40	695	Pentium 800
	3659.10	2960.50	2719.00	1834.10	3276.00	2548.70	169 968	1 run, 54.4 min

Table 4
Comparison of results for the 400-customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	Cumulative	CPU
Gehring and Homberger (1999)	36.40	8.00	38.00	12.00	36.10	8.60	1390	Pentium 200
	8925	6502	7584	3935	8763	5518	412 270	1 run, 420 min
Gehring and Homberger (2001)	36.40	8.00	38.00	12.00	36.10	8.80	1392	Pentium 400
	9478.22	6650.28	7855.82	3940.19	9294.99	5629.43	428 489	3 runs, 47.1 min
Li and Lim (2003)	36.60	8.00	38.70	12.10	36.50	9.50	1414	Pentium 545
	8912.40	6610.60	7181.40	4017.10	8377.90	5466.20	405 656	3 runs, 359.8 min
Bräysy et al. (2004a, 2004b)	36.40	8.00	37.90	12.00	36.00	8.90	1391	AMD 700
	8692.17	6382.63	7230.48	3894.48	8305.55	5407.87	399 132	3 runs, 7.9 min
Bent and Van Hentenryck (2004)	36.40	8.00	38.00	12.00	36.10	8.90	1393	Sun Ultra 10
	8713.37	6959.75	7220.96	4154.40	8330.98	5631.70	410 112	N/A, N/A
Homberger and Gehring (2005)	36.40	8.00	38.10	12.00	36.10	9.20	1397	Pentium 400
	9547.86	6683.53	7921.19	4049.71	9296.75	5609.88	431 089	3 runs, 5.1 min
Le Bouthillier and Crainic (2005)	36.50	8.00	37.90	12.00	36.00	8.60	1390	Pentium 933
	8839.28	6437.68	7447.09	3940.87	8652.01	5511.22	408 281	1 run, 520 min
Mester and Bräysy (2005)	36.30	8.00	37.90	12.00	36.00	8.80	1389	Pentium 2000
	8530.03	6209.94	7148.27	3840.85	8066.44	5243.06	390 386	1 run, 17 min
Pisinger and Röpke (2005)	36.40	8.00	37.60	12.00	36.00	8.50	1385	Pentium 3000
	8540.04	6241.72	7290.16	3844.69	8069.30	5335.09	393 210	5 runs, 15.8 min
MP-ES	36.30	8.00	38.00	12.00	36.10	8.90	1390	Pentium 800
	8648.20	6317.70	7182.60	3862.70	8192.20	5258.80	394 818	1 run, 238.5. min

and Pisinger and Röpke (2005) present solutions with a lower total number of routes (405 instead of 406), but at the expense of a higher total distance. Here one must note that the results of Pisinger and Röpke (2005) are the best over 10 test runs while the reported computing time is the average for a single test run. For an updated list of best-known solutions for the VRPTW, the reader is referred to www.top.sintef.no.

For the 200 and 400 customer benchmarks, MP-ES improves total distance by 1.41% and 3.30% respectively, compared to the previous best approaches generating the same cumulative number of routes on the benchmarks. The eight new best solutions found for these benchmarks are listed in Table 5.

In addition to 17 new best-known solutions, the MP-ES algorithm found a solution that matches the current best-known solution to 51 tested VRPTW instances. Overall, and regardless of problem size, the computational efforts required by MP-ES appear to be higher compared to other methods. However, here one should bear in mind that MP-ES is implemented with Visual Basic and we use a dynamic graphical user interface (see Fig. 2) that clearly reduces the program speed compared to other methods.

Table 5
17 new best-known solutions

Problem	Vehicles	Total distance	Problem	Vehicles	Total distance
R104	9	1007.24	C2-2-3	6	1775.11
R106	12	1251.98	C2-2-10	6	1806.60
R110	10	1118.59	RC2-2-2	5	2827.45
R203	3	939.54	C1-4-1	40	7152.06
R208	2	725.75	C1-4-5	40	7152.02
R210	3	939.34	C1-4-6	40	7153.41
RC201	4	1406.91	C2-4-1	12	4116.05
RC204	3	798.41	C2-4-7	12	3894.13
RC205	4	1297.19			

3.4. Results for the real-life problems

The results for the four real-life problems reported in Taillard (1993) and Fisher (1994) are described in Table 6. Taillard's problem T385 was generated by selecting the most important towns or villages in a canton of Vaud in Switzerland and by basing the demands on the number of inhabitants in each commune. As the name implies, there are 385 customers in the problem. The problems F44 and F134 of Fisher represent a day of grocery deliveries in Peterboro and Bramalea Ontario terminals whereas F71 is concerned with the delivery of tires, batteries and accessories to Exxon gasoline stations. For each problem both the total distance and CPU time in minutes is reported. The computer used and the number of computational test runs is given in the rightmost column.

According to the table, MP-ES found a match with the best-known solution for three of the four problems. For T385 the obtained solution is about 2.2% worse than the best-known one. The computation effort required by MP-ES is competitive with the other methods listed in Table 6.

Results for the two real-life problems of Russell (1995) are compared in Table 7. The problems are extracted from a fast food delivery application in the South Eastern United States. There are 417 customers in both problems and the problems differ only in that the problem E417 has a higher percentage of tight delivery time windows.

As one can see from the table, only Mester and Bräysy (2005) report a better solution than MP-ES to the two problems. MP-ES dominates other previously published solution methods and taking into account the differences

Table 6
Comparison of results for the real-life problems of Taillard (1993) and Fisher (1994)

Reference	T385		F44		F71		F134		Computer, simulations
	Dist.	CPU	Dist.	CPU	Dist.	CPU	Dist.	CPU	
Taillard (1993)	24599.60	520	–	–	–	–	–	–	Silicon Graphics 35, 1 run
Fisher (1994)	–	–	723.54	N/A	241.97	N/A	1163.60	N/A	Apollo Domain 3000, 1 run
Rochat and Taillard (1995)	24435.50	N/A	–	–	–	–	1162.96	N/A	Siligon Graphics Indigo, –
Xu and Kelly (1996)	–	–	723.54	85.8	244.54	5199	1176.72	11490	DEC Alpha 1 run
Mester and Bräysy (2005)	24855.32	840	723.54	0.08	241.97	2.0	1162.96	24.0	Pentium 2000, 1 run
MP-ES	24965.20	9000.0	723.54	0.1	241.97	0.3	1162.96	10.0	Pentium 800, 1 run

Table 7
Comparison of results for the two real-life problems by Russell (1995)

Reference	Problem D417		Problem E417		CPU
	Vehicles	Distance	Vehicles	Distance	
Kontoravdis and Bard (1995)	55	4273.40	55	4985.70	Sun Sparc 10, 5 runs, 11 min
Rochat and Taillard (1995)	54	6264.80	54	7211.83	Silicon Graphics 100 MHz, -,-
Russell (1995)	55	4964.00	55	6092.00	PC 486 66 MHz, 3 runs, 7 min
Taillard et al. (1997)	55	3439.80	55	3707.10	Sun Sparc 10, -,-
Chiang and Russell (1997)	55	3455.28	55	3796.61	Pentium 166 MHz, -, 37 min
Liu and Shen (1999)	54	3747.52	54	4691.14	HP 9000/720, 3 runs, 45 min
Homberger and Gehring (1999)	54	4703.00	55	4732.00	Pentium 200 MHz, 5 runs, 30 min
Gehring and Homberger (2001)	54	3512.01	54	3832.24	4 × Pentium 400 MHz, 5 runs, 142 min
Bräysy (2003)	54	3506.21	54	3801.64	Pentium 200 MHz, 1 run, 378 min
Bräysy et al. (2004a, 2004b)	54	3387.93	54	3672.73	AMD 700 MHz, 3 runs, 47 min
Mester and Bräysy (2005)	54	3317.19	54	3542.73	Pentium 2000, 1 run, 167 min
MP-ES	54	3369.04	54	3660.43	Pentium III 800 MHz, 1 run, 265 min

in the speeds of the used computers, MP-ES is significantly faster than the heuristic of Mester and Bräysy (2005).

4. Conclusion

The capacitated vehicle routing problems with and without time windows constitute one of the classical areas in Operation Research and have considerable economic significance. They have many real-life applications, especially in the transportation industry. For industrial problems, scalable methods that are able to produce high quality results in limited time, even for several hundreds of customers, are particularly important. Our solution algorithm is based on the ideas of $(1 + 1)$-evolution strategies and a new multi-parametric mutation procedure based on the ruin and recreate principle. The extensive computational experiments on six real-life problems and 199 standard benchmark problems demonstrate that the suggested algorithm is efficient and competitive with the state-of-art solution methods from the literature. In 42% of the test problems our algorithm either matched or improved the best-known solution. We would also like to note that an earlier version suggested solution method has successfully been applied to solve real (50–150 customer) vehicle routing problems up to 100–150 times a day within the European project SFB559—The modeling of Large Networks in Logistics, in the part A11—Redistribution Networks.

Acknowledgements

This work was partially supported by the Ministry of Absorption of Israel, the MINERVA Optimization Laboratory (Technion), the Foundation for Economic Education, and the EDGE project funded by the Research Council of Norway. This support is gratefully acknowledged. We would like to thank also A. Ben-Tal, A. Nemirovsky, A. Korol, and E. Taillard for their valuable feedback.

References

Bent, R., & Van Hentenryck, P. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science, 38*, 515–530.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing, 15*, 347–368.

Bräysy, O., Dullaert, W., & Gendreau, M. (2004b). Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics, 10*, 587–611.

Bräysy, O., Hasle, G., & Dullaert, W. (2004a). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research, 159*, 586–605.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science, 39*, 104–118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science, 39*, 119–139.

Chiang, W. C., & Russell, RA. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing, 9*, 417–430.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial optimization* (pp. 315–338). Chichester: Wiley.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*, 512–522.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, JS. (2005). New heuristics for the vehicle routing problem. In A. Langevin & D. Riopel (Eds.), *Logistic systems, design and optimization* (pp. 279–297). New York: Springer.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum K-trees. *Operations Research, 42*, 626–642.

Flood, M. M. (1956). The travelling-salesman problem. *Operations Research, 4*, 61–75.

Gehring, H., & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99* (pp. 57-64). Jyväskylä, University of Jyväskylä.

Gehring, H., & Homberger, J. (2001). Parallelization of a two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research, 18*, 35–47.

Gendreau, M., Laporte, G., & Potvin, J.-Y. (2001). Metaheuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 129–154). Philadelphia: SIAM.

Homberger, J., & Gehring, H. (1999). Two evolutionary meta-heuristics for vehicle routing problem with time windows. *INFOR, 37*, 297–318.

Homberger, J., & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research, 162*, 220–238.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., & Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science, 39*, 206–232.

Kontoravdis, G. A., & Bard, JF. (1995). A GRASP for the vehicle routing problem with time windows. *INFORMS Journal on Computing, 7*, 10–23.

Laporte, G., & Semet, F. (2001). Classical heuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 109–128). Philadelphia: SIAM.

Le Bouthillier, A., & Crainic, TG. (2005). Cooperative parallel method for vehicle routing problems with time windows. *Computers and Operations Research, 32*, 1685–1708.

Li, H., & Lim, A. (2003). Local search with annealing-like restarts to solve the VRPTW. *European Journal of Operational Research, 150*, 115–127.

Liu, F.-H., & Shen, S.-Y. (1999). A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research, 118*, 485–504.

Mester, D., & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers and Operation Research, 32*, 1593–1614.

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Ph.D. thesis, Department of Industrial Engineering and Management Science, North Western University, USA.

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Annals of Operation Research, 41*, 421–451.

Pisinger, D., & Röpke, S. (2005). *A general heuristic for vehicle routing problems*. Technical Report, Department of Computer Science, University of Copenhagen, Denmark.

Potvin, J., & Rousseau, J. (1995). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society, 46*, 1433–1446.

Rechenberg, I. (1973). *Evolutionstrategie*. Stuttgart: Fromman-Holzboog.

Rochat, Y., & Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics, 1*, 147–167.

Russell, R. A. (1995). Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science, 29*, 156–166.

Schwefel, H.-P. (1977). *Numeriche optimierung von computer-modellen mittels der evolutions-strategie*. Basel: Birkauser.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operation Research, 35*, 254–265.

Taillard, E. (1993). Parallel iterative search methods for vehicle routing problems. *Networks, 23*, 661–673.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science, 31*, 170–186.

Xu, J., & Kelly, J. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science, 30*, 379–393.