Available online at www.sciencedirect.com

Computers & Operations Research 34 (2007) 2964–2975

www.elsevier.com/locate/cor

Active-guided evolution strategies for large-scale capacitated vehicle routing problems

David Mester[a,*], Olli Bräysy[b]

[a]*Mathematical and Population Genetics Laboratory, Institute of Evolution, University of Haifa, 31905 Haifa, Israel*
[b]*Agora Innoroad Laboratory, Agora Center, University of Jyväskylä, P.O. Box 35, FI-40014, Finland*

Available online 13 December 2005

Abstract

We present an adaptation of the active-guided evolution strategies metaheuristic for the capacitated vehicle routing problem. The capacitated vehicle routing problem is a classical problem in operations research in which a set of minimum total cost routes must be determined for a fleet of identical capacitated vehicles in order to service a number of demand or supply points. The applied metaheuristic combines the strengths of the well-known guided local search and evolution strategies metaheuristics into an iterative two-stage procedure. The computational experiments were carried out on a set of 76 benchmark problems. The results demonstrate that the suggested method is highly competitive, providing the best-known solutions to 70 test instances.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Vehicle routing; Heuristics; Evolution strategies; Guided local search

1. Introduction

Logistics, and especially the distribution of goods, lies at the heart of business activity. Formally, most problems in the domain of goods distribution can be viewed as vehicle routing problems (VRP). The capacitated VRP (CVRP) constitutes the classical version of the VRP and assuming the symmetry of the cost matrix, we define the problem formally on an undirected graph $G = (V, E)$ where $V = \{v_0, \ldots, v_n\}$ is the vertex set and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is an edge set. Vertex v_0 represents a depot, while the remaining vertices correspond to customers. With each vertex $v_i \in V$ is associated a non-negative demand q_i and a service time s_i. With each edge (v_i, v_j) is associated a non-negative cost, c_{ij}, interpreted here as a travel time. All vehicles have identical capacity, Q, and the number of vehicles is not determined a priori. The CVRP consists in determining a set of vehicle routes (a) starting and ending at the depot, and such that (b) each customer is visited exactly once, (c) the total demand of any vehicle route does not exceed Q, (d) the duration of any route (including service times) does not exceed a (possibly) preset upper limit, and (e) the total cost of all routes is minimized.

Here, one must note that in the literature the above described VRP with route duration limit is often separated from CVRP and called distance-constrained VRP. The algorithms presented in this paper are designed for and used to solve problems both without and with route duration limit. As the VRP is a very complex NP-hard problem, solving the VRP to optimality is not always possible within the limited computing time one often has in practical situations. In

* Corresponding author.
E-mail address: dmester@research.haifa.ac.il (D. Mester).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.006

these cases one must focus on heuristic and metaheuristic solution methods that will produce high-quality solutions in limited time. For more details, we refer to extensive surveys of Laporte et al. [1], Laporte and Semet [2], Gendreau et al. [3] and Cordeau et al. [4,5]. For exact solution methods, we refer to surveys by Toth and Vigo [6], Naddef and Rinaldi [7] and Bramel and Simchi-Levi [8].

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 adaptation of the active-guided evolution strategies (AGES) metaheuristic of Mester and Bräysy [9] for the CVRP. Our experiments on 76 CVRP benchmarks from the literature demonstrate that the proposed algorithm is fast, cost-effective and highly competitive. It finds the best-known solutions to 70 out of the 76 tested CVRP instances.

The remainder of this paper is outlined as follows. The main features of the AGES metaheuristic are described in Section 2. Section 3 details the parameter values of the tested algorithm configurations and presents the results of a comprehensive computational study. Finally, in Section 4 conclusions are drawn.

2. The problem solving methodology

The AGES metaheuristic consists of two phases. The goal of the first phase is to create a starting solution for the second phase. The starting solution is generated by creating first a set of s solutions with the hybrid cheapest insertion heuristic of Mester et al. [10] and then selecting the best solution found as the starting solution. The exact value of s depends on the chosen algorithm configuration. After the starting solution has been created, the filling procedure of Bent and Van Hentenryck [11] is optionally applied to reduce the number of routes in the starting solution before proceeding to the second phase. The usage of the filling procedure depends on the chosen algorithm configuration as detailed in Section 3.1.

In the second phase an attempt is made to improve the starting solution with a two-stage procedure. The first stage makes use of the well-known guided local search (GLS) metaheuristic [12,13]. GLS operates by augmenting the objective function with a penalty term based on particular solution features (e.g. long edges) not considered to be part of a near-optimal solution. The GLS is used to guide a composite local search procedure, consisting of 3–5 different improvement heuristics, described in more detail in the next subsection. When no more improvements have been found for a given number of iterations, the second stage is started. The second stage operates by performing a series of removal and reinsertion operations as in the large neighborhood search of Shaw [14]. As the removal and reinsertion operations follow the principles of the $1 + 1$ evolution strategies (ES) metaheuristic [15,16], the second stage is called ES-stage. The second stage is continued until no more improvements can be found and then the search goes back to the first stage. The two stages are repeated iteratively, and the search is stopped if no more improvements can be found by neither stages. We call the configuration presented in Mester and Bräysy [9] for the CVRP with time window constraints the *AGES VRPTW best* configuration. In this paper we propose two alternative configurations for CVRP, noted as *AGES VRP best* and *AGES VRP fast*.

2.1. The improvement heuristics

Before proceeding to a description of the global heuristic, we first recall the main features of the improvement heuristics that it will control and guide. Four of the available six heuristic procedures (forward Or-exchange, backward Or-exchange [17], 1-interchange [18] and 2-opt* [19]) are used for inter-route improvements only, i.e., they modify two routes simultaneously. 2-opt [20] works on a single route at a time and relocate [21] is used for both inter- and intra-route reinsertions.

The basic idea of the relocate and forward and backward Or-exchange procedures is to reinsert a 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. Relocate examines all possible insertion positions between two consecutive customers both in the current route of the customer considered for relocation and in alternate routes. 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

Table 1
The AGES configurations

Config.	α	s	v	BVH	λ	$c_{\max}$[a]	PVN	$l_{\text{PVN}}^{\text{lower}} - l_{\text{PVN}}^{\text{upper}}$	RS: ρ; M; α	Composite local search
AGES	0.6–1.4	5	20	Yes	0.05	25	25–∞	3 routes–	1: 1; 7; 0.2–1.4[b]	Relocate, 1-interchange,
VRPTW	1.4							$0.6 \times n$ customers	2: 0.5; 7; 0.2–1.4[c]	2-opt*
best									3: 0.5; 7; 0.2–1.4[c]	
AGES	1.0	1	20	No	0.01	20	25–∞	2 routes–	1: 0.05; 3; 0.8–1.2	Forward Or-exchange,
VRP								$(0.2 + 0.5 \times a^2) \times n$	2: 0.95; 5; 0.6–1.4	Backward Or-exchange,
best								customers	3: 0	1-Interchange, 2-opt, 2-opt*,[d]
AGES	1.0	1	20	No	0.01	20[a]	25–200	3 routes always	1: 0.01; 3; 0.8–1.2	Forward Or-exchange,
VRP									2: 0.99; 5; 0.6–1.4	Backward Or-exchange,
fast									3: 0	2-opt[e]

[a]The second stage is terminated either after first improvement is found or if no more improvements can be found.
[b]Applied only within the last seven iterations; the search is repeated for 14 iterations before evaluating the solutions.
[c]Applied randomly within the first seven iterations out of a total of 14, see Mester and Bräysy [9] for details.
[d]2-opt* is applied here only with 50% probability.
[e]In the AGES VRP fast configuration, the heuristics in the composite local search are executed only once instead of repeating them in a loop.

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 origin route. 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.

Depending on the chosen algorithm configuration, 3–5 of the above described improvement heuristics are selected to a set called composite. The heuristics in the composite are repeated in a loop until no more improvement can be found such that after each successful move the heuristic is changed. This composite procedure is called composite local search in the remaining of the paper. The details of the applied composite local search configurations are given in Section 3.1.

2.2. The initial solution phase

The starting solution for the second phase is created with the heuristic of Mester et al. [10] that hybridizes a cheapest reinsertion procedure with the composite local search. 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 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 $k_{\text{p}} < l_{\text{p}} < m_{\text{p}} \leqslant n$ 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 limitation of the insertion positions is used to avoid moving customers back and forth in the solution vector and to facilitate reducing the number routes. All feasible reinsertions are evaluated using the modified insertion criterion of Osman [18]

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

where α is a parameter whose values are detailed in Table 1 (Section 3.1) and the reinsertion with the maximum positive value is executed (reinsertions with negative value of the criterion (1) are disregarded). The first term in the criterion refers to the costs of the edges to be deleted in the considered insertion and the second term to new edges to be created in the insertion of k to a new position. Costs of the edges related to old position of k (c_{ik}, c_{kj}, c_{ij}) are positive whereas costs of the edges related to the considered new position of k (c_{lk}, c_{km}, c_{lm}) are negative. Each time v% of the customers have been reinserted, the composite local search is applied until no improvement can be found. The reinsertions are repeated until no more routes can be eliminated from the solution. To be more precise, the search is stopped if the algorithm cannot find a route whose all customers can be inserted in alternative routes. By varying the value of α, several different initial solutions can be created. In the end, the best initial solution found with different values of α is selected as the starting solution for the second phase.

2.3. The improvement phase

In the second phase an attempt is made to improve the created starting solution with an iterative two-stage local search procedure. The first stage uses a simple GLS to guide the composite local search. The GLS works by penalizing always only a single edge every time the algorithm gets stuck in a local minimum. The edge chosen for penalization is the edge in the current solution for which the highest value of the following "utility" function is achieved:

$$U = \frac{c_{ij}}{(1 + p_{ij})}, \tag{2}$$

where p_{ij} is the penalty counter for edge (i, j) which holds the number of times the edge has been penalized. When evaluating the moves within the composite local search, the new cost c_{ij}^* of the currently penalized edge is calculated using Eq. (2):

$$c_{ij}^* = c_{ij} + p_{ij}\lambda L, \tag{3}$$

where L is the average length of edges in the starting solution and λ is a parameter. To avoid calculating the penalized costs many times, they are stored in a matrix. After determining the edge to be penalized, the next step is to define so-called penalty variable neighborhood (PVN) that is used to restrict the composite local search to the geographically closest routes to the currently penalized edge to speed up the calculations in large-scale problems. For more guidelines for solving large-scale routing problems, we refer to Kytöjoki et al. [22]. To be more precise, the PVN is formed by selecting first the route in which the penalized edge (i, j) is. Then, all customers are sorted according to how close they are to customers i and j, and all the remaining routes are ranked according to how close customers they serve. The second route included in the PVN is the one serving the closest customer to i. Correspondingly, the third route is the one serving the closest customer to j, and the fourth route is the one serving the second closest customer to i and so on. The current size of the PVN l_{PVN} is set randomly within given limits $l_{\text{PVN}}^{\text{lower}} - l_{\text{PVN}}^{\text{upper}}$ in the beginning of every iteration

$$l_{\text{PVN}} = l_{\text{PVN}}^{\text{lower}} + (l_{\text{PVN}}^{\text{upper}} - l_{\text{PVN}}^{\text{lower}})a^2, \tag{4}$$

where a is a random number between 0 and 1. The limits depend on the chosen AGES configuration, as detailed in Section 3.1. The applied limits are based on both the number of routes and the number of customers in the selected routes.

If no more improvements have been found for a user-defined number of iterations $c_{\max}^1$ and the current PVN size is within defined limits (see Section 3.1) the second stage is started. The basic idea of the second ES-stage is to remove a selected set of customers from the current solution and then reinsert the removed customers. Two or three strategies are applied randomly to select the customers to be removed, depending on the algorithm configuration. The alternative three removal strategies (RS) are:

Strategy 1: Select all customers n_{PVN} in the current PVN.

Strategy 2: Select $(0.2 + 0.5a)n_{\text{PVN}}$ customers randomly from the current PVN (a is a random value uniformly distributed between 0 and 1).

Strategy 3: Select a random number of customers from the current PVN within a ring created by two circles with random radiuses (in terms of the used cost unit such as Euclidean distance), centered at the depot.

The probabilities ρ for applying the alternative removal strategies are detailed in Section 3.1. After the removal, the selected customers are reinserted using the cheapest insertion heuristic of Mester et al. [10] with the composite local search and the same GLS objective function as in the first stage. Each time after all removed customers have been inserted, an attempt is made to improve the obtained solution with the GLS composite local search of the first stage. Each entity of removing and reinserting a set of customers and improving the obtained solution with the GLS composite local search is called an iteration. Depending on the chosen algorithm configuration and applied removal strategy (see Table 1 in Section 3.1), the search is repeated for a given number of iterations M in the second stage with different values of α for the heuristic of Mester et al. [10] before evaluating the created solutions and replacing the current parent solution in case a better solution is found according to the GLS-based objective function. If the algorithm cannot improve the current solution at all in the second stage, the currently penalized edge is determined randomly instead of heuristic rule based on Eq. (1) in the first stage to diversify the search.

In case of a large PVN (more than 2 routes), the PVN is decomposed in the second stage into sets of two different routes. After the decomposition each pair of routes make up their own PVN. All possible combinations of these two-route PVNs are then considered in a loop that is restarted from the first pair every time an improvement is found. The second stage is continued until no more improvements can be found. Then the search goes back to the first stage in case one of the two stages has found improvements. Otherwise, the search is terminated.

As deteriorating is allowed, the algorithm maintains in memory the best solution found during the entire search and returns that solution in the end. The best solution found is used also as a restart point if no improvements have been found for $c^2_{\max}$ iterations in neither stage. The value of $c^2_{\max}$ is defined randomly between 5000 and 50 000. In the restart all penalty counters p_{ij} are reinitialized to zero. For more details and a detailed algorithm, we refer to Mester and Bräysy [9] and Mester et al. [23].

3. Experimental results

In this section we describe the alternative algorithm configurations, the test data sets and present sensitivity analysis and the results of the suggested AGES metaheuristic along with a comparative analysis with the state-of-art methods from the literature.

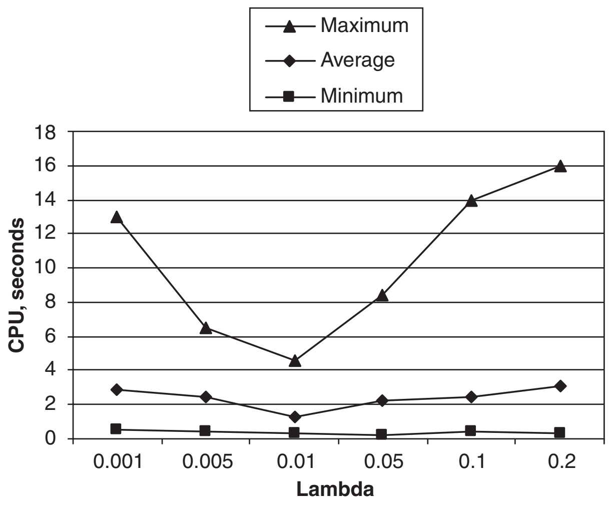
3.1. The algorithm configurations

To summarize the differences between the previously published AGES VRPTW algorithm and the two alternative configurations of AGES for CVRP, presented in this paper, we have congregated in Table 1 the parameters of the AGES algorithm and their values in different configurations. Parameters α, s and v are used in the cheapest insertion heuristic to control the value of insertion criterion, the number of initial solutions created and the frequency of improvements with the composite local search during the construction, respectively. The values of parameter α are always varied in increments of 0.2 units within the given ranges in the table. Notation BVH refers to whether or not the filling procedure of Bent and Van Henrenryck is applied and λ controls the penalties in the GLS-based objective function in the improvement phase. Parameter $c^1_{\max}$ sets the limit for non-improving iterations in the first improvement stage to launch the second improvement stage and PVN refers to the limit set for the PVN size in terms of the number of customers to enter the second stage. The range $l^{\text{lower}}_{\text{PVN}} - l^{\text{upper}}_{\text{PVN}}$ gives the minimum and maximum size for the PVN and a is a random number distributed uniformly between 0 and 1. The removal strategy parameters ρ and M define the probabilities of selecting the three alternative removal strategies and the number of iterations the search is repeated in the second stage. In the composite local search, the operators are applied in the same order as they are listed in the table.

3.2. Problem data and the experimental setting

The proposed algorithm has been implemented in Visual Basic 6.0 and the algorithm was executed on a Pentium IV Net Vista PC2800 MHz (512 Mb RAM) computer. The computational tests were carried out with the standard CVRP benchmarks of Christofides et al. [24], Golden et al. [25] and Li et al. [26] and with the Gehring and Homberger [27] test instances for the VRPTW. The 14 classical benchmark problems of Christofides et al. [24] can be divided both in random (problems 1–10) and clustered problems (problems 11–14) and in problems without (problems 1–5 and 11–12) and with maximum tour length constraint (problems 6–10 and 13–14). The 20 large-scale benchmarks of Golden et al. [25] consist of 200–483 customers. The first eight instances have route-length restrictions. Each problem has a geometric structure: eight problems have customers located in concentric circles around the depot, four problems have customers located in concentric squares with the depot located in one corner, four problems have customers located in concentric squares around the depot, and four problems have customers located in a six-pointed star around the depot. The problems of Li et al. [26] have 560–1200 customers and route-length restrictions, and they exhibit geometric structure (concentric circles around the depot).

Gehring and Homberger suggested in total 300 test instances, divided in six groups and five different problems sizes. Within each group the problems differ only according to their time windows. As we do not consider time window constraints in this paper, we solved only one problem without time windows from each group, i.e., 30 problems. Travel



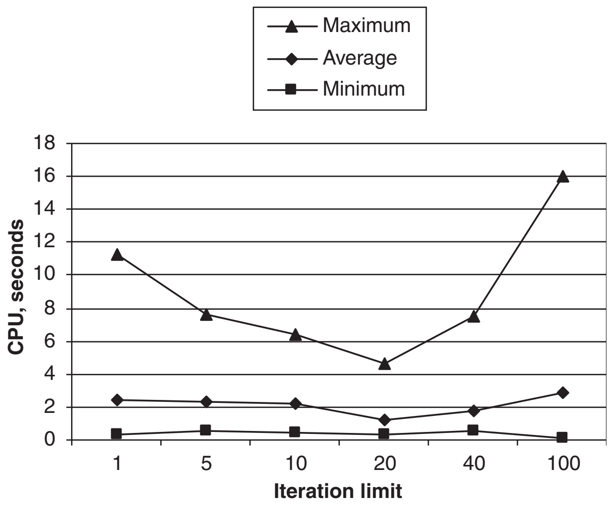


Fig. 1. The sensitivity of CPU time with respect to parameters λ and $c^1_{\max}$.

Table 2
The significance of the initial solution

Method	Initial distance	Time to 2%		Distance after 60 s
		s	σ	
Construction only	1648.67	2.7	2.6	1102.33
Construction with composite local search	1288.83	1.9	1.9	1103.00

times between two customers correspond to their relative Euclidean distance in all problems. The experimental tests consist only of one simulation run for each problem.

The sensitivity of the results to the parameter values described in the previous subsection were examined using the problems 9–20 of Golden et al. [25]. In the sensitivity analysis the values of other parameters than the one under examination were always fixed to their best values, detailed in Table 1. As the main focus is on large-scale problems, special emphasis in the sensitivity analysis and in the selection of final parameter values was put on speed. Fig. 1 illustrates the sensitivity of the computing time with respect to the values of parameter λ and iteration limit $c^1_{\max}$ of the first stage that controls the balance of the two stages. The three curves represent the longest, average and shortest CPU time observed over the problems 9–20 of Golden et al. [25].

According to the figure, the differences between the average and minimum CPU times appear very small. Values 0.01 and 20 appear the best for λ and $c^1_{\max}$, respectively, and therefore they were used in all other experimental tests.

The significance of the initial solution is illustrated in Table 2, and also in Table 3. The second column of Table 2 describes the average total distance of the 12 starting solutions (for problems 9–20) created with the insertion heuristic of Mester et al. [10] without and with the composite local search. The third column gives the average computing time in seconds to reach solution quality within 2% from the best-known and the standard error for the computing time. The last column shows the average distance over the 12 problems after computing time of 60 s, given the two different initial solutions.

It is obvious that applying the composite local search within the initial insertion heuristic improves significantly the quality of the starting solution, as one can see from Table 2. It appears also that the algorithm reaches faster the 2% solution quality limit by using starting solutions of better quality. The same conclusions are supported also by the results presented in Table 3. On the other hand, the solution quality after the 60 s computing time appears the same regardless of the solution quality of the starting solution.

Table 3
Comparison among the applied improvement operators within the composite local search

Applied improvement operators					Initial distance (%)	Time to 2%		Distance after 60 s (%)
Forward Or-exchange	Backward Or-exchange	1-Interchange	2-opt	2-opt*		s	σ	
Yes	No	No	No	No	49.03	94.5	101.3	6.72
No	No	No	Yes	No	32.63	72.1	101.4	4.61
No	No	No	Yes	Yes	31.32	14.5	18.3	3.73
Yes	No	No	Yes	No	24.00	5.5	7.8	2.37
Yes	No	Yes	No	No	26.80	26.9	45.5	2.00
Yes	Yes	No	No	No	44.36	8.6	6.9	1.28
Yes	No	Yes	No	Yes	21.86	12.4	20.3	2.14
Yes	No	No	Yes	Yes	22.67	6.1	7.7	2.03
Yes	Yes	Yes	No	No	25.47	5.7	5.1	1.47
Yes	Yes	No	Yes	No	15.35	1.5	1.2	0.72
Yes	No	Yes	Yes	Yes	17.60	3.2	3.9	1.71
No	Yes	Yes	Yes	Yes	17.95	4.5	5.1	1.64
Yes	Yes	No	Yes	Yes	19.54	2.0	2.3	1.10
Yes	Yes	Yes	Yes	No	18.37	2.6	3.2	0.97
Yes	Yes	Yes	No	Yes	20.93	3.4	3.6	0.62
Yes	Yes	Yes	Yes	Yes	17.59	1.9	1.9	0.64

Table 3 presents the efficiency of the composite local search within AGES with different combinations of the improvement heuristics. The same composite local search is applied in both phases of the AGES algorithm. The first five columns describe whether or not the operator in question is applied. The initial distance column gives the average difference in percents between the starting solutions and the best-known solutions for the 12 test problems given the usage of the marked improvement heuristics during the construction of the starting solution. The next column presents the average computing time in seconds to reach solution quality (in terms of distance) that is within 2% from the best-known and the standard error of the computing time. The last column shows the average percentual difference in the solution quality after 1 min CPU time with respect to the best-known solutions.

In general, it appears that by applying more operators one gets better final results according to Table 3 and according to the initial distance column, the quality of the starting solution is also better if more operators are applied in the construction. The quality of the starting solution certainly affects also the efficiency of the composite local search in the improvement phase and the quality of the final solutions but based on Table 3 conclusions are hard to make as same operators are applied in both phases in all cases. The forward and backward Or-exchange appear most important operators and 2-opt the least valuable, although all five operators are important and contribute to the solution quality. The best efficiency and solution quality are achieved by applying all five operators. It is interesting note that it is possible to get quite competitive results by applying only forward and backward Or-exchange operators. The combination of forward and backward Or-exchange with 2-opt appears also very efficient. With some combinations such as forward Or-exchange, 2-opt and 2-opt* there is a conflict between the results presented in the last two columns: for example in the above case it takes 6.1 s to reach the 2% solution quality limit. On the other hand, the percentual difference to the best solutions after 60 s is still over 2% (2.03). The reason is that AGES always prefers a solution with a smaller number of routes. In this case the number of routes has been reduced between the two examination points.

In Table 4 we compare the two configurations of AGES (AGES VRP best and fast) suggested in this paper for CVRP against the configuration presented in Mester and Bräysy [9] for VRP with time windows. In addition, Table 4 illustrates the significance of the two AGES stages by showing results obtained with GLS-stage and ES-stage only (without the other stage). As in previous tables, the comparison is based on problems 9–20 of Golden et al. [25]. To make possible fair comparison, all configurations started from the same initial solution as AGES VRP best and all configurations were terminated after same CPU time limit, given in the last column. The time limit was determined using the actual computing time of AGES VRP best configuration (after which it could not find any more improvements).

Table 4
Comparison between the AGES VRP and VRPTW configurations and two stages of AGES using problems 9–20 of Golden et al. [25]

Problem	AGES VRP best	AGES VRP fast	AGES VRPTW best	GLS-stage	ES-stage	CPU (s)
9	583.39	585.99	585.32	589.26	591.13	360.2
10	741.56	749.94	741.76	749.91	754.64	75.0
11	918.45	926.53	929.53	932.56	936.47	440.8
12	1107.19	1117.64	1022.81	1135.82	1128.40	647.7
13	859.11	859.86	866.22	867.11	880.07	400.0
14	1081.31	1092.59	1106.37	1101.58	1106.58	48.5
15	1345.23	1354.08	1363.97	1355.62	1366.65	27.6
16	1622.69	1621.46	1626.12	1638.88	1649.48	800.1
17	707.79	708.02	710.18	710.48	710.35	30.2
18	998.73	1002.81	997.52	1008.11	1008.45	150.6
19	1366.86	1376.75	1390.02	1380.44	1376.28	23.3
20	1820.09	1821.17	1843.74	1835.39	1836.56	230.0
Average dev. %	0.00	0.49	0.24	1.16	1.46	

According to Table 4 the best solution quality is obtained with the AGES VRP best configuration. For some individual problems AGES VRP fast and AGES VRPTW best configurations found slightly better solutions than AGES VRP best, but they could not achieve the same average solution quality with equal computing time, which illustrates the efficiency of the AGES VRP best configuration. In general, the differences between the three AGES configurations are however small, indicating the flexibility of the algorithm. According to the table, it appears that to reach the best solution quality both AGES stages are clearly required.

3.3. Results for the Christofides et al. benchmark problems

The results for the 14 CVRP instances of Christofides et al. [24] are presented in Table 5. We consider in Table 5 only a limited set of results of recent metaheuristics that have shown the best performance according to the literature. For a more thorough comparison, we refer the reader to Gendreau et al. [3], Cordeau et al. [4] and Tarantilis [28]. The first row lists the authors, and the next 14 rows show the total distance values of each author for the 14 problems. The last three rows report the average deviation from the best-known solutions in percentage, type and speed in MHz of the computers used, and the average CPU time per problem in minutes. The first two columns give the problem number and size and the number of vehicles in our solutions. The best-known solution values are taken from www.top.sintef.no and given in the rightmost column. Our results and computation times in seconds over a single test run are shown in the AGES VRPTW best, AGES VRP fast, AGES VRP best and CPU/s columns. Here, one must note that AGES VRPTW best uses a different code than the other two configurations. AGES VRPTW best was designed for time-constrained problems without route-length restrictions. Therefore, it is used to solve only problems without route duration limit.

According to Table 5, the suggested method is competitive. It found the best-known solution to all problems, except to problem 10. The average deviation from the best-known solutions is only 0.03% and 0.07% for the AGES VRP best and fast configurations, respectively. Our method appears to be competitive also in terms of computation time. Only Toth and Vigo [30] is faster than AGES VRP fast configuration, even when the differences in computer speeds are taken into account, using e.g. the factors of Dongarra [32]. It is noteworthy that in case of problems 7 and 8 AGES VRP best configuration is faster than the AGES VRP fast configuration, and finds very quickly the best-known solution.

3.4. Results for the large-scale benchmark problems of Golden et al.

In Table 6 we compare the results for the 20 large-scale benchmark instances of Golden et al. [25]. The format of the table is the same as for Table 5, and as in Table 5, we consider here only a limited set of the most recent and most competitive results. Our solutions are marked in bold in cases where we obtained a new best-known solution.

In Table 6, we see that the suggested AGES metaheuristic found the best-known solution to all problems except two (problems 6 and 7). To 12 problems AGES VRP best found new best-known solution whereas AGES VRPTW best and

Table 5
The results for the capacitated vehicle routing benchmarks of Christofides et al. [24]

Problem	Veh.	Rochat and Taillard [29]	Toth and Vigo [30]	Reimann et al. [31]	Tarantilis [28]	AGES VRPTW best	CPU (s)	AGES VRP fast	CPU (s)	AGES VRP best	CPU (s)	Previous best known
1 (50)	5	524.61	524.61	524.61	524.61	524.61	1.9	524.61	0.1	524.61	0.2	524.61
2 (75)	10	835.26	838.60	840.61	835.26	835.26	44.1	835.26	3.8	835.26	5.5	835.26
3 (100)	8	826.14	828.56	828.21	826.14	826.14	81.3	826.14	0.7	826.14	1.0	826.14
4 (150)	12	1028.42	1033.21	1037.57	1028.42	1028.42	51.5	1028.42	6.6	1028.42	10.2	1028.42
5 (199)	16	1291.45	1318.25	1306.91	1311.48	1291.29	6116.0	1294.25	9.6	1291.29	2160	1291.29
6 (50)	6	555.43	555.43	553.43	555.43	–	–	555.43	0.1	555.43	4.2	555.43
7 (75)	11	909.68	920.72	917.50	909.68	–	–	909.68	7.2	909.68	0.8	909.68
8 (100)	9	865.94	869.48	865.94	865.94	–	–	865.94	6.6	865.94	0.8	865.94
9 (150)	14	1162.55	1173.12	1173.94	1162.55	–	–	1164.54	1.4	1162.55	25.8	1162.55
10 (199)	18	1395.85	1435.74	1415.53	1407.21	–	–	1404.67	1.4	1401.12	52.2	1395.85
11 (120)	7	1042.11	1042.87	1043.46	1042.11	1042.11	4.0	1042.11	0.7	1042.11	1.1	1042.11
12 (100)	10	819.56	919.56	819.56	819.56	819.56	2.2	819.56	0.2	819.56	0.2	819.56
13 (120)	11	1541.14	1545.51	1546.84	1544.01	–	–	1543.26	2.5	1541.14	13.5	1541.14
14 (100)	11	866.37	866.37	866.37	866.37	–	–	866.37	1.2	866.37	1.7	866.37
Dev. %		0.01	0.64	0.48	0.18	N/A		0.07		0.03		0.00
Computer		SGI[a] 100	Pentium 200	Pentium 900	Pentium 400	Pentium IV 2800		Pentium IV 2800		Pentium IV 2800		N/A
CPU min.		N/A	3.8	3.8	6.6	N/A		0.05		2.8		N/A

[a]SGI = Silicon Graphics Indigo.

Table 6
Comparison of results for the large-scale benchmarks of Golden et al. [25]

Problem	Veh.	Reimann et al. [31]	Kytöjoki et al. [22]	Pisinger and Röpke [33]	AGES VRPTW best	CPU (s)	AGES VRP fast	CPU (s)	AGES VRP best	CPU (s)	Previous best known
1 (240)	9	5644.02	5867.84	5650.91	–	–	5644.00	4.3	**5627.54**	187.7	5639.36
2 (320)	10	8449.12	8476.26	8469.32	–	–	8468.00	5.3	8447.92	1016.1	8447.92
3 (400)	10	11 036.22	11 043.41	11 047.01	–	–	11 146.00	3.1	11 036.22	860.5	11 036.22
4 (480)	10	13 699.11	13 631.72	13 635.31	–	–	13 704.52	8.3	13 624.52	21 421.9	13 624.52
5 (200)	5	6460.98	6460.98	6466.68	–	–	6460.98	6.3	6460.98	76.3	6460.98
6 (280)	7	8412.90	8415.67	8416.13	–	–	8539.61	2.0	8412.88	1636.2	8412.80
7 (360)	9	10 195.59	10 297.66	10 181.75	–	–	10 240.42	7.3	10 195.56	56.0	10 181.75
8 (440)	10	11 828.78	11 872.64	11 713.62	–	–	11 918.75	3.2	**11 663.55**	735.7	11 696.55
9 (255)	14	586.87	620.67	585.14	**583.39**	605.0	588.43	55.2	**583.39**	360.2	585.43
10 (323)	16	750.77	784.77	748.89	**741.56**	423.0	752.92	5.6	**741.56**	75.0	743.17
11 (399)	18	927.27	986.80	922.70	922.18	1337.0	925.94	3.3	**918.45**	440.8	923.11
12 (483)	19	1140.87	1209.02	1119.06	1011.42	2720.0	1120.67	110	**1107.19**	647.7	1116.22
13 (252)	26	865.07	925.81	864.68	**859.11**	730.0	865.20	0.8	**859.11**	400.0	862.38
14 (320)	30	1093.77	1155.19	1095.40	1082.47	807.0	1097.68	4.1	**1081.31**	48.5	1083.65
15 (396)	33	1358.21	1461.49	1359.94	1347.19	577.0	1354.76	16.8	**1345.23**	27.6	1351.35
16 (480)	37	1635.16	1742.86	1639.11	1622.69	3240.0	1634.99	20	1622.69	800.1	1632.47
17 (240)	22	708.76	726.01	708.90	**707.79**	42.0	710.22	0.7	**707.79**	30.2	708.63
18 (300)	27	998.83	1077.53	1002.42	**997.52**	150.0	1009.53	5.2	998.73	150.6	998.83
19 (360)	33	1367.20	1444.51	1374.24	**1366.86**	526.0	1381.88	6.2	**1366.86**	23.3	1367.20
20 (420)	38	1822.94	1938.12	1830.80	1821.87	940.0	1840.57	2.1	**1820.09**	230.0	1822.94
Dev. %		0.32	1.63	0.15	N/A		0.80		−0.08		0.00
Computer		Pentium 900	Athlon64 3000+	Pentium IV 3000	Pentium IV 2800		Pentium IV 2800		Pentium IV 2800		N/A
CPU min.		49.3	0.02	10.8	N/A		0.2		24.4		N/A

Table 7
Computational results for the 12 benchmark problems of Li et al. [26]

Problem	Vehicles	Li et al. [25]	Kytöjoki et al. [32]	Pisinger and Röpke [33]	AGES VRP fast	CPU (s)	AGES VRP best	CPU (s)	Previous best known
21 (560)	10	16 602.99	16 221.22	16 224.81	16 602.99	2.0	**16 212.74**	179.2	16 212.83
22 (600)	15	14 651.27	14 654.87	14 631.08	14 652.44	13.1	**14 597.18**	35.1	14 631.08
23 (640)	10	19 005.37	18 810.72	18 837.49	18 810.72	4.6	**18 801.12**	40 000.8	18 801.13
24 (720)	10	21 784.43	21 401.41	21 522.48	21 401.41	5.8	**21 389.33**	100.4	21 389.43
25 (760)	20	17 151.43	17 358.18	16 902.16	17 358.18	9.0	17 095.27	122.4	16 902.16
26 (800)	10	24 189.66	23 996.86	24 014.09	23 996.86	13.9	**23 971.74**	130.0	23 977.74
27 (840)	20	17 823.40	18 233.93	17 613.22	17 823.40	17.8	**17 488.74**	293.2	17 613.22
28 (880)	10	26 606.11	26 592.05	26 791.72	26 605.13	31.9	**26 565.92**	20 000.2	26 566.04
29 (960)	10	29 181.21	29 166.32	29 405.60	29 166.32	20.4	29 160.33	150.3	29 154.34
30 (1040)	10	31 976.73	31 805.28	31 968.33	31 976.73	16.1	**31 742.51**	900.9	31 742.64
31 (1120)	10	35 369.17	34 352.48	34 770.34	35 366.74	19.4	**34 330.84**	5000.5	34 330.94
32 (1200)	10	37 421.44	37 025.37	37 377.35	37 364.10	56.2	36 928.70	8175.1	36 919.24
Dev. %		1.22	0.48	0.63	1.00		0.015		0.00
Computer		Athlon 1000	Athlon64 3000+	Pentium IV 3000	Pentium IV 2800		Pentium IV 2800		N/A
CPU min.		3.2	0.1	49.8	0.3		104.3		N/A

AGES VRP fast found new best-known solution to one problem (see also Table 4). The average deviation of the results obtained with the AGES VRP best configuration from the previous best-known solutions is −0.08%, which is better than for any other method. The computational effort required by AGES VRP best configuration is, however, higher than for most of the other methods considered in Table 6. Only Pisinger and Röpke [33] can be considered slower given that the results of Pisinger and Röpke [33] are the best over 10 test runs while the reported computing time is the average for a single test run. Here, one must also that note that our Visual Basic implementation with graphical user interface is significantly slower compared to optimized C + + code. When the fast configuration is used, only Kytöjoki et al. [22] remains faster than AGES but AGES is outperformed by all other authors in terms of solution quality. According to Table 6, it seems that for some problems such as problem 4, an exceptionally high computation is required for AGES VRP best. The reason for this is the stopping criterion where the search is continued as long as the algorithm can find improvements. In some problems AGES VRP best seems to be able to continue improving the solutions slightly for a very long time.

3.5. Results for the very large-scale benchmark problems of Li et al. [26]

The results for the 12 very large-scale instances of Li et al. [26] are presented and compared against the results reported in literature in Table 7.

As can be seen from the table, the suggested metaheuristic found a new best-known solution to nine problems with the best configuration. The average deviation of the AGES VRP best configuration from the previous best-known solutions is 0.015%, which is better than for the previously suggested methods. On the other hand, the computation times of AGES with the best configuration are higher than for the other competing methods, especially in case of problems 23 and 28. With the fast configuration, the computation times are a lot shorter but the results are outperformed by Kytöjoki et al. [22] and Pisinger and Röpke [33] in terms of solution quality. The differences remain, however, small.

3.6. Results for the large-scale VRPTW benchmark problems of Gehring and Homberger

In this section we describe our experimental results for the 30 large-scale VRPTW instances of Gehring and Homberger [27]. We are not aware of any other research on these problems without considering the time window constraints. Therefore, we present in Table 8 the number of vehicles and total distance for each problem obtained with the AGES algorithm only. We solved each problem only once using the AGES VRP fast configuration and CPU time limit of 25 min.

Table 8
The results for the Gehring and Homberger [27] extended VRPTW instances

Problem	Vehicles	Distance	Problem	Vehicles	Distance
C1_200	18	2570.49	R2_200	4	1610.30
C1_400	36	6765.03	R2_400	8	3323.05
C1_600	56	13 465.24	R2_600	11	6254.17
C1_800	72	23 999.35	R2_800	15	10 230.06
C1_1000	90	39 811.15	R2_1000	19	15 149.40
C2_200	4	1368.89	RC1_200	18	2804.20
C2_400	8	2799.40	RC1_400	36	7381.47
C2_600	12	5441.98	RC1_600	55	14 786.15
C2_800	16	8241.83	RC1_800	73	26 720.23
C2_1000	20	11 792.66	RC1_1000	90	41 583.54
R1_200	18	2878.24	RC2_200	4	1513.00
R1_400	36	7192.03	RC2_400	8	3100.06
R1_600	54	15 691.37	RC2_600	11	5732.08
R1_800	72	27 650.67	RC2_800	15	9217.99
R1_1000	92	42 311.91	RC2_1000	18	13 631.76

By comparing the results presented in Table 8 with the best-known solutions reported in SINTEF web page (www.top.sintef.no) for the Gehring and Homberger benchmarks with time windows, it is interesting to note that the results are close to each other, except for problem group C2 to which the time window-constrained solutions are about 50% worse.

4. Conclusions

In this paper we have adapted the AGES metaheuristic of Mester and Bräysy [9] to the CVRP. In the first phase a starting solution is generated with the hybrid cheapest insertion heuristic of Mester et al. [10]. In the improvement phase, the two powerful metaheuristics GLS and ES are combined into iterative two-stage procedure. The suggested solution method has been tested on 76 test problem instances from the literature. The results show that the proposed metaheuristic is efficient and competitive in comparison to the previous heuristic solution methods, providing the best-known solutions to 70 of the tested 76 VRPs within reasonable CPU times.

Acknowledgements

This work was partially supported by the Ministry of Absorption of Israel, the Foundation for Economic Education, Henry Ford and Volvo Foundations, and the EDGE project funded by the Research Council of Norway. This support is gratefully acknowledged. We would like to thank also Wout Dullaert for his valuable feedback.

References

[1] Laporte G, Gendreau M, Potvin J-Y, Semet F. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operations Research 2000;7:285–300.

[2] Laporte G, Semet F. Classical heuristics for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia: SIAM; 2001. p. 109–28.

[3] Gendreau M, Laporte G, Potvin J-Y. Metaheuristics for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia: SIAM; 2001. p. 129–54.

[4] Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y, Semet F. A guide to vehicle routing heuristics. Journal of the Operational Research Society 2002;53:512–22.

[5] Cordeau J-F, Gendreau M, Hertz A, Laporte G, Sormany JS. New heuristics for the vehicle routing problem. In: Langevin A, Riopel D, editors. Logistics systems: design and optimization. New York: Springer; 2005. p. 279–97.

[6] Toth P, Vigo D. Branch-and-bound algorithms for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia: SIAM; 2001. p. 29–52.

[7] Naddef D, Rinaldi G. Branch-and-cut algorithms for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia: SIAM; 2001. p. 53–84.

[8] Bramel J, Simchi-Levi J. Set-covering-based algorithms for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia: SIAM; 2001. p. 85–108.

[9] Mester D, Bräysy O. Active guided evolution strategies for the large scale vehicle routing problem with time windows. Computers & Operations Research 2005;32:1593–614.

[10] Mester D, Bräysy O, Dullaert W. A multi-parametric evolution strategies algorithm for vehicle routing problems. Working paper, University of Haifa, Israel, 2005.

[11] Bent R, Van Hentenryck P. A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science 2004;38:515–30.

[12] Voudouris C. Guided local search for combinatorial problems. PhD dissertation, University of Essex, UK, 1997.

[13] Voudouris C, Tsang E. Guided local search. European Journal of Operations Research 1998;113:80–119.

[14] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget J-F, editors. Principles and practice of constraint programming—CP98, lecture notes in computer science. New York: Springer; 1998. p. 417–31.

[15] Rechenberg I. Evolutionsstrategie. Stuttgart: Fromman-Holzboog; 1973.

[16] Schwefel H-P. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. Basel: Birkhäuser; 1977.

[17] Or I. Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. PhD thesis, Northwestern University, Evanston, USA, 1976.

[18] Osman IH. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. Annals of Operations Research 1993;41:421–52.

[19] Potvin J-Y, Rousseau J-M. An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society 1995;46:1433–46.

[20] Flood MM. The traveling-salesman problem. Operations Research 1956;4:61–75.

[21] Savelsbergh MWP. The vehicle routing problem with time windows: minimizing route duration. INFORMS Journal on Computing 1992;4: 146–54.

[22] Kytöjoki J, Nuortio T, Bräysy O, Gendreau M. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Computers & Operations Research, in press, doi: 10.1016/j.cor.2005.10.010.

[23] Mester D, Korol A, Nevo E. Fast and high precision algorithms for optimization in large scale genomic problems. Computation Biology and Chemistry 2004;28:281–90.

[24] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. Chichester: Wiley; 1979. p. 315–38.

[25] Golden BL, Wasil EA, Kelly JP, Chao IM. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Boston: Kluwer; 1998. p. 33–56.

[26] Li F, Golden B, Wasil E. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research 2005;32:1165–79.

[27] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. In: Proceedings of EUROGEN99. Jyväskylä: University of Jyväskylä; 1999. p. 57–64.

[28] Tarantilis CD. Solving the vehicle routing problem with adaptive memory programming methodology. Computers & Operations Research 2005;32:2309–27.

[29] Rochat Y, Taillard E. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1:147–67.

[30] Toth P, Vigo D. The granular tabu search (and its application to the vehicle routing problem). INFORMS Journal on Computing 2003;15: 333–48.

[31] Reimann M, Doerner K, Hartl RF. D-ants: savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research 2004;31:563–91.

[32] Dongarra J. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Department of Computer Science, University of Tennessee, USA, 2005.

[33] Pisinger D, Röpke S. A general heuristic for vehicle routing problems. Computers & Operations Research, in press, doi: 10.1016/j.cor.2005.09.012.