Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

An evolution strategy approach to the team orienteering problem with time windows

Korhan Karabulut[a,*], M. Fatih Tasgetiren[b]

[a] Software Engineering Department, Yasar University, Selcuk Yasar Campus, 35100 Izmir, Turkey
[b] Mechanical and Industrial Engineering Department, Qatar University, Doha, Qatar

ARTICLE INFO

Keywords:
Evolution strategy
Team orienteering problem with time windows
Ruin and recreate heuristic

ABSTRACT

The team orienteering problem with time windows (TOPTW) is a highly constrained NP-hard problem having many practical applications in vehicle routing and production scheduling. The TOPTW is an extended variant of the Orienteering Problem (OP), where each node has a predefined time window during which the service has to be started. The aim is to maximize the total collected score by visiting a set of nodes with a limited number of tours since the given distance budget is limited. We propose an evolution strategy (ES) together with an effective constructive heuristic for solving the TOPTW. The main feature of the ES is to generate an offspring solution through ruin and recreate (RR) heuristic, where a number of nodes are removed from the incumbent solution and then, they are reinserted into tours until a complete solution is obtained. The ES is hybridized with an efficient random local search to enhance solution quality. For survivor selection, we use a goodness of scores approach to determine and diversify the population for the next generation. Parameters of the ES are determined through the design of experiment approach to tune them. The computational results show that the constructive heuristic is slightly better than existing heuristics in the literature. Furthermore, the detailed computation results on the benchmark suite from the literature confirm the effectiveness of the evolution strategy. Ultimately, the evolution strategy obtains new best-known solutions for 7 benchmark problem instances.

1. Introduction and literature review

As a sports discipline, orienteering can be defined as follows. Given a set of control points with associated profits together with the start and end points, the orienteering problem (OP) aims to obtain a tour between the start and endpoints so as to maximize the total profit collected; subject to a given distance budget. Since distance is limited, a tour cannot include all points. The first public orienteering competition was held in 1897 in Norway; however, the practice in the army is earlier (IOF, n.d.).

Orienteering was first introduced as an optimization problem by Tsiligirides (1984). Then, Golden, Levy, and Vohra (1987) presented a heuristic algorithm to solve the OP in three steps: route construction, route improvement, and center of gravity improvement. Golden, Wang, and Liu (1988) designed a new and improved heuristic for the OP. The randomization concept of Tsiligrides is embedded in the center of gravity procedure together with learning capabilities. Keller (1989) proposed an algorithm for a multi-objective vending problem to solve the OP. His algorithm consists of two stages: path construction and improvement. Ramesh and Brown (1991) developed an efficient four-phase heuristic for the OP. Their heuristic consists of vertex insertion, cost improvement, vertex deletion, and maximal insertions. Chao (1993) and Chao, Golden, and Wasil (1996) developed a fast and effective heuristic for the OP. Tasgetiren and Smith (2000) and Tasgetiren (2002) developed genetic algorithms for the OP.

The OP can be described as a combination of the 0–1 knapsack and the TSP (Vansteenwegen, Souffriau, & Oudheusden, 2011). Certain problems in logistics, tourism and other areas can be modeled as an orienteering problem (Vansteenwegen et al., 2011). There exist several variants of the OP in the literature. One version is called the OP with time windows (OPTW), where a time window refers to the practical situation that control points should be visited within a predefined time interval to start the service. This additional constraint makes finding feasible solutions harder. The literature on the OPTW is very scarce. Kantor and Rosenwein present a heuristic based on exhaustive search (1992), Mansini, Pelizzari, and Calvo (2006) have designed a granular variable neighborhood search, and Righini and Salani (2009) proposed a dynamic programming approach. Santini (2019) proposed a heuristic that uses clustering and an algorithm based on adaptive large neighborhood search.

* Corresponding author.
E-mail addresses: korhan.karabulut@yasar.edu.tr (K. Karabulut), mtasgetiren@qu.edu.qa (M.F. Tasgetiren).

https://doi.org/10.1016/j.cie.2019.106109
Received 9 April 2019; Received in revised form 24 June 2019; Accepted 1 October 2019
Available online 26 October 2019

Another variant is the team orienteering problem (TOP), which is proposed by Chao et al. (1996a, 1996b). In the TOP, a fixed number m of paths is considered with a given time budget for each path. Chao et al. (1996a, 1996b) employed a two-phase heuristic algorithm, originally designed for the OP (Chao, 1993) to solve the TOP. A tabu search is presented by Tang and Miller-Hooks (2005) to solve the TOP. An ant colony optimization (ACO) algorithm is proposed by Ke, Archetti, and Feng (2008). A memetic algorithm is presented by Bouly, Dang, and Moukrim (2010). Souffriau, Vansteenwegen, Berghe, and Van Oudheusden (2010) develop a path-relinking algorithm combined with a greedy randomized adaptive search procedure. Vansteenwegen et al. also present a guided local search and a skewed variable neighborhood search for the TOP (2009a, 2009b, 2009c).

The TOP with time windows (TOPTW) is an extended variant of the TOP, where each node has a predefined time window during which the service has to be started. The aim is to maximize the total collected score by visiting a set of nodes with a limited number of tours since the given distance budget is limited. Since the OP is NP-hard (Golden et al., 1987), it is unlikely to solve the TOPTW to optimality within polynomial time. All solution approaches for the TOPTW in the literature are metaheuristics, except for a few exact methods. An ant colony system (ACS) designed to solve the hierarchic generalization of the TOPTW is presented by Montemanni and Gambardella (2009). It is further enhanced by Montemanni, Weyland, and Gambardella (2011) by developing enhanced ACS (EACS). The EACS algorithm proposes two more components to handle the drawbacks of ACS. Both components employ the best solution found so far in the construction phase ASC and local search is applied to those solutions on which local search has not been recently applied. An iterated local search (ILS) algorithm is developed by Vansteenwegen et al. (2009a, 2009b, 2009c), which can provide fast solutions to the TOPTW. The proposed ILS is very simple and effective in a way that the current solution is perturbed and insertion based local search is applied to the perturbed solution. Tricoire, Romauch, Doerner, and Hartl (2010) developed a variable neighborhood search (VNS) algorithm with several neighborhood structures to solve the multi-period orienteering problem with multiple time windows, which is another generalization of the TOPTW, where visits are extended to multiple time-periods instead of a single period. A hybrid method consisting of a greedy randomized adaptive search procedure (GRASP) with evolutionary local search (ELS) is presented by Labadie, Melechovský, and Calvo (2011). Several constructive heuristics based on GRASP are developed for establishing the initial population. Then, those solutions in the initial population are further enhanced by ELS. Lin and Yu (2012) developed simulated annealing (SA)-based two heuristics called “slow SA (SSA)” and “fast SA (FSA)”. FSA is aimed at getting quick responses, whereas SSA is developed for getting high-quality solutions at the expense of more CPU time. Labadie, Mansini, Melechovský, and Calvo (2012) developed a linear programming-based granular variable neighborhood search algorithm that can explore fewer neighborhoods instead of complete ones, without losing the effectiveness of the algorithm. Souffriau, Vansteenwegen, Vanden Berghe, and Van Oudheusden (2011) developed a hybrid GRASP and ILS algorithm named GRILS to newly proposed multi constraint team orienteering problem with multiple time windows as well as the TOPTW and selective vehicle routing problem with time windows. Hu and Lim (2014) developed an iterative framework including three components, a local search procedure, a simulated annealing procedure and a route recombination method (I3CH). Artificial bee colony (ABC) approach is first presented by Karabulut and Tasgetiren (2013). Later on, Cura (2014) also developed an ABC algorithm, where the hybridization of SA and a new scout bee search behavior based on a local search procedure are introduced to improve the solution quality of benchmark instances. Ke, Guo, and Zhang (2014) present an approach that uses both a metaheuristic and a branch-and-price algorithm whereas Gedik, Kirac, Milburn, and Rainwater (2017) presented a constraint programming (CP) approach to tackle the TOPTW. Very recently, Gunawan, Lau, Vansteenwegen, and Lu (2017) improved the GRILS approach and present two new algorithms, i.e. iterated local search (ILS), and a hybrid Simulated Annealing ILS (SAILS) that are executed using different computation times in order to compare to the state-of-the-art algorithms. Gavalas, Konstantopoulos, Mastakas, and Pantziou (2019) proposed clustering-based heuristics to extend the ILS algorithm that could improve solution quality with respect to ILS. Yu, Jewpanya, Lin, and Redi (2019) used a hybrid ABC to solve TOPTW-TDS which extends TOPTW by adding time-dependent scores where the profit of a location is dependent on the time of visit. They also report results of their algorithm for the TOPTW instances. A multi-objective TOPTW is proposed by Hu, Fathi, and Pardalos (2018) where different kinds of profits can be assigned to each checkpoint.

Two exact methods for the OPTW (a special case of the TOPTW with only one tour) using bi-directional dynamic programming are presented by Righini and Salani (2006, 2009) that can find optimal results for the OPTW instances. To date, the only exact algorithm for the TOPTW, which is based on a branch-and-price approach, is given by Tae and Kim (2015) that can solve some of the TOPTW benchmark instances to optimality. A survey is first presented in Vansteenwegen et al. (2011). More details about the performances of the state-of-the-art algorithms for the TOPTW can be found in a recent survey in Gunawan, Lau, and Vansteenwegen (2016). In addition, practical applications of the TOPTW can be found in Souffriau, Vansteenwegen, Vertommen, Berghe, and Oudheusden (2008), Vansteenwegen and Oudheusden (2007), and Wang, Golden, and Wasil (2008).

For the description of the TOPTW, we follow Labadie et al. (2012) as follows. Suppose that a set of nodes to be visited $N = \{1, 2, ..,n\}$ is given with a depot denoted by 0. Tours always start and end at the depot. Assume that a copy of the depot is also indexed by $n + 1$. A directed graph can be defined as $G = (N \cup \{0, n + 1\}, E)$, where $N \cup \{0, n + 1\}$ is the set of nodes and E is a set of edges connecting the nodes such that $E = \{(i, j): i \neq j \in N \cup \{0, n + 1\}\}$. Each node $i \in N$ has a profit P_i and each node can be visited at most once, whereas there is no profit for visiting the depot, i.e., P_0 and $P_{n+1} = 0$. Each node $i \in N$ has a time window $[e_i, l_i]$, in which e_i is the earliest visiting time and l_i is the latest visiting time allowed to start the service at node$i$. On the other hand, e_0 and l_0 are the earliest visiting time each tour to depots 0 and $n + 1$. There exist m tours, each limited by the maximum tour length T_{max}. We assume that $e_0 = e_{n+1} = 0$ and $l_0 = l_{n+1} = T_{max}$. There is a cost c_{ij} in time units for travelling between nodes i and j and each node i has a service time denoted by S_i. The problem is to find m tours with a maximum total profit. Each tour starts at time e_0, visits a subset of nodes exactly once within their time windows and ends at the depot $n + 1$ before the time $l_0 = l_{n+1} = T_{max}$.

Let L_i be the departure time from node i after its service is finished. With $L_0 = 0$, L_i can be calculated as follows:

$$L_i = \max(L_{i-1} + c_{i-1,i}, e_i)+S_i, \tag{1}$$

Let A_i be the arrival time at node i that can be calculated as follows:

$$A_i = L_{i-1} + c_{i-1,i}. \tag{2}$$

Then, the start of service time for node i can be calculated as follows:

$$SST_i = \max(L_{i-1} + c_{i-1,i}, e_i) \tag{3}$$

A feasible solution to the TOPTW requires that the service for node i should start before its latest visiting time ($SST_i \leq l_i$) and arrival time to depot should be smaller than or equal to the maximum tour length ($A_{n+1} \leq T_{max}$) for all m tours. By the definitions above, a feasibility check of a solution can be easily carried out when making any move in any tour. Note that we only consider feasible moves in this study.

In this paper, an effective constructive heuristic, which is denoted as KT and an evolution strategy (ES) are presented for the TOPTW. The KT heuristic is compared to best-known or optimal solutions in the literature and shown to be very effective and efficient in finding solutions to the existing problem instances. The proposed ES approach obtained new best-known solutions for 7 problem instances for the first time in the literature.

The rest of the paper is organized as follows: Section 2 presents the KT heuristic, Section 3 gives details of the ES algorithm for the TOPTW, and the computational experiments with parameter setting are given in Section 4. Finally, the conclusion is given in Section 5.

2. The constructive heuristic

Constructive heuristics aim to develop good solutions/individuals by starting from an empty solution and iteratively adding solution components to the current partial solution. Since the TOPTW is a constrained optimization problem, a constructive heuristic must ensure that the solution is feasible when new components are added to the partial solution. Constructive heuristics generally use a problem-specific cost function to select the next candidate node to insert into the partial solution. One such function is given by Vansteenwegen et al. (2009a, 2009b, 2009c). Three insertion heuristics and two heuristics based on sweep algorithm to construct initial solutions are developed by Labadie et al. (2011).

The insertion move tries to add one more node to the current solution. When a new node is considered for insertion into one of the current tours, it should be checked whether the inserted node and the nodes after the insertion point have their services started before their latest service starting time, in order to ensure that the solution remains feasible. If not, the insertion of the node will be infeasible. When making insertions, a ranking method based on the cost function is used to select the next node from the unvisited list of nodes. Insertion of a new node may increase both travel costs and profits or vice versa. A straightforward approach is to select a node that will add more profit with less travel cost. Since insertion neighborhood typically makes insertion of all unvisited nodes in all positions of all tours of a solution, it consumes substantial computation time. If the insertion of a node into a particular position in a particular tour is feasible, the imposed increase in the total travel cost can be calculated faster as explained by Vansteenwegen et al. (2009a, 2009b, 2009c) as follows:

The speed-up method for feasibility check for insertion can be implemented by keeping track of waiting times (w_i) and maximum delay time for the start of service (d_i) for all visited nodes. w_i is the waiting time at node i before the service starts. It is positive if the arrival time is before the earliest visiting time or 0 otherwise: $w_i = \max(e_i - A_i, 0)$. d_i is the maximum possible delay time for node i, which is either limited by its own latest visiting time or by the next node's in the permutation, i.e., $d_i = \min(l_i - SST_i, w_{i+1} + d_{i+1})$.

The travel time increase when an unvisited node j is inserted between nodes i and $i+1$ can be calculated as:

$$\Delta_j = c_{i,j} + w_j + S_j + c_{j,i+1} - c_{i,i+1} \tag{4}$$

For a feasible insertion, Δ_j should be less than or equal to the sum of waiting time and maximum possible delay for node $i+1$:

$$\Delta_j \leq w_{i+1} + d_{i+1} \tag{5}$$

A "desirability measure" can be calculated for each feasible insertion. Vansteenwegen et al. (2009a, 2009b, 2009c) employed an insertion method that all nodes are inserted in all tours, and chose the node with the highest P_j^2/Δ_j ratio. They state that travel time increase becomes less important than profit because of the time windows, therefore profit is squared. They also report that worse results are obtained when profit is not squared. We denote this heuristic as VI insertion heuristic. Another measure is presented in Labadie et al. (2011), which is $q_j \bullet P_j/\Delta_j$, where q_j is the number of reachable neighbors (these are the neighbors that can be visited without a constraint violation) from node j. A candidate node with more reachable neighbors will have a higher priority, assuming that this scheme will increase the probability of finding other feasible insertions in the next iterations.

The above measure is used in two insertion heuristics in Labadie et al. (2011). The first heuristic, denoted as LI1, starts from the first tour and considers the insertion of all unvisited nodes only to the last position of the current tour. The node with the highest $q_j \bullet P_j/\Delta_j$ ratio is selected for insertion. If there is no feasible insertion to the current tour, the next tour is considered. This process continues until m tours are constructed. In the second heuristic, denoted as LI2, all nodes are sorted according to their profits in a decreasing order and m nodes with the highest profits are inserted into m tours. Then, all unvisited nodes are considered for insertion into all positions in all tours. Again, the insertion position and the unvisited node that have the highest ratio are selected. Another heuristic, denoted as LI3 is proposed by Labadie et al. (2011). LI3 starts with sorting all nodes according to their profits in a decreasing order and considers the node at the top of the list for insertion into all positions in all tours. The insertion with a minimum Δ_j is selected. If there is no feasible insertion for the current node, the next node in the unvisited list is considered. The heuristic stops when all nodes are considered for insertion. In addition to the above, two more heuristics (LS1 and LS2) are also proposed by Labadie et al. (2011). They are adaptions of the well-known sweep heuristic for vehicle routing problems.

Profits and increases in travel costs for visited nodes should be carefully considered for constructing good solutions and have been considered in heuristics proposed by Vansteenwegen et al. (2009a, 2009b, 2009c) and Labadie et al. (2011). Another important point is the latest visiting time. The nodes with sooner latest visiting times would be harder to add to the partial solutions later. The constructive heuristic proposed in this study takes the latest visiting time of a candidate node into account, in addition to the profit and cost increase associated with the node. The proposed heuristic uses a ratio calculated for each feasible insertion as follows:

$$e^{-(l_j/l_{max})} \bullet P_j^2/\Delta_j \tag{6}$$

where l_{max} is the maximum latest visiting time. The heuristic uses the ratio of the squared additional profit divided by delta travel time increase developed by Vansteenwegen et al. (2009a, 2009b, 2009c). Since the problem includes time windows, the latest visiting time of the possible new node to be inserted is also important. For two alternative insertions with equal or close P_j^2/Δ_j ratios, the node with sooner latest visiting time should be chosen so that other feasible insertions could be possible in later iterations of the heuristic. The proposed ordering scheme favors the nodes with sooner latest visiting times and insertions with the smaller cost for the same node. Note that throughout the paper, we use this desirability measure in all parts of the ES algorithm wherever necessary when making feasible insertions. The computational experiments presented in Section 4 show that the proposed KT heuristic generates slightly better results on overall average when compared to the other six heuristics developed by Vansteenwegen et al. (2009a, 2009b, 2009c) and Labadie et al. (2011).

3. Evolution strategy approach

Evolution strategy (ES) is a metaheuristic optimization algorithm developed by Rechenberg (1971) and Schwefel (1975). ES is typically applied to real parameter optimization problems and it uses mutation for offspring generation. The mutation is performed by adding a random value drawn from a normal distribution. Parameter tuning is an important issue when using evolutionary algorithms, including ES. The idea of evolving parameters within the individuals, self-adaption, is considered as a standard approach in ES (Beyer & Schwefel, 2002). The main assumption in evolving parameters within individuals is; a good parameter value that generates a good individual survives together with that individual.

$$
\begin{aligned}
&rS \leftarrow ruin\ size\\
&T_1 \leftarrow \{2,4,8\}\\
&T_2 \leftarrow \{1,5,7\}\\
&U \leftarrow \{3,6,9,10\}\\
&TC_1 \leftarrow c_{02} + c_{24} + c_{48} + c_{80}\\
&TC_2 \leftarrow c_{01} + c_{15} + c_{57} + c_{70}\\
&TTC \leftarrow TC_1 + TC_2\\
&f(\lambda) \leftarrow P_2 + P_4 + P_8 + P_1 + P_5 + P_7
\end{aligned}
$$

Fig. 1. Solution representation.

3.1. Solution representation

Solution representation for the self-adaptive ES uses one permutation for each tour and one list for the unvisited nodes and the ruin size parameter value rS that is the number of nodes that will be removed from the solution in the ruin and recreate procedure. The details of the ruin and recreate procedure will be presented later. In the solution representation, each solution employs m partial permutations in order to represent m tours. Each tour starts and ends at the depot, which is node 0. However, the depot is not included in the permutations. Let T_k be the kth tour, then, a solution can be represented by $T_k \in (k = 1, .., m)$. Nodes that are not present in any of the tours are stored in U, which is the list of unvisited nodes. Note that c_{ij} is the cost of the distance between nodes i and j. Let TC_k be the cost of the tour T_k. In the solution representation, we also keep the objective function value (fitness) of all m tours as $f(\lambda)$, which is the total collected profit for all m tours, as well as the total travel cost of the solution, $TTC = \sum_{k=1}^{m} T_k$. For example, given $N = \{1, 2, .., 10\}$ and $m = 2$, Fig. 1 illustrates the solution representation:

3.2. Initial population

Three individuals in the initial population with size $|\lambda|$ are constructed using three of the constructive heuristics: LS1, VI and KT. We use only these heuristics, not the other four, because these three produce better results and take less CPU time when compared to the other heuristics. For the rest of the initial population, one of these three individuals are selected randomly, and is perturbed by applying two neighborhood structures randomly selected out of four neighborhood structures used in the local search procedure that is explained in section 3.4. The local search procedure given in Algorithm 1 is also applied to all individuals in the initial population and the best so far individual is recorded. The evolution loop is repeated until the predefined time limit or an optimum solution is reached. The number of offspring generated in each iteration is seven times the population size (Eiben & Smith, 2003).

3.3. Offspring population

Parent selection for reproduction is done by using a discrete uniform distribution, $DU(1, |\lambda|)$, thus, each individual has an equal chance for selection. Once an individual λ_i from the parent population $|\lambda|$ is selected randomly, the offspring individual μ_i is generated by using the ruin and recreate (RR) algorithm (Schrimpf, Schneider, Stamm-Wilbrandt, & Dueck, 2000). RR is a simple but effective method, in which a number of nodes are removed from the current solution and added to the solution again in a way that it optimizes the chosen criterion. It is shown to generate very good offspring solutions for combinatorial optimization problems by Schrimpf et al. (2000). The single and very important parameter to be adjusted for the RR algorithm is the ruin size, which is the number of nodes to be removed from the solution. A small value does not create a move that is large enough to escape local optima, while a large value destroys a good solution. As stated before, a self-adaptive scheme can be used instead of presetting such a strategy parameter. The ruin size for the offspring is calculated by mutating the current ruin size as $rS_i = \lfloor rS_i.\ e^{\tau \cdot N(0,1)} \rfloor$, where τ is the learning rate, $N(0, 1)$ is a random number generated from a unit normal distribution with mean zero and standard deviation one. The ruin size is mutated by the unit normal distribution, $N(0, 1)$, that leads to a small variation with a high probability and a large variation with a low probability, in order to obtain the offspring ruin size. If the offspring ruin size is outside the predetermined limits, it is restricted to the predetermined levels.

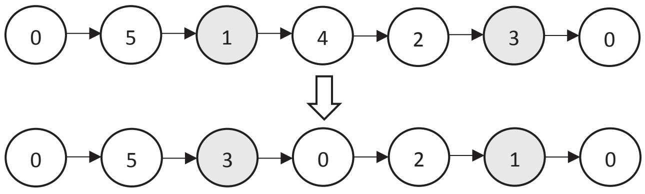

Fig. 3. An example of N_2.

In the ruin part of the RR algorithm, a number rS_i of nodes are removed from the tours as follows: For each node to be removed, a random tour is selected from individual λ_i, then, a single node is removed from the tour and the node is added to the list U of unvisited nodes. This is repeated for rS_i times.

In the recreate part, we use the neighborhood structure, $N_0 = MoveFromUnvisitedList$, which is inspired from the multiple insertion heuristic (MIH) proposed for solving parallel machine scheduling problem with sequence-dependent setup times in Kurz and Askin (2001). This neighborhood structure aims at increasing the profits of tours, hence the fitness of a solution. Simply, each node in the unvisited list U is inserted into all positions in all tours in the hope of increasing the profits with an additional feasible node. Note that the neighborhood structure N_0 is computationally very expensive because of inserting a node to every possible position in m tours. In order to accelerate it, we take advantage of the nearest neighbor approach for insertions in Snyder and Daskin (2006). Suppose that there is a tour T_k with total cost TC_k and node x will be inserted into an edge $\{i, j\}$. The cost of the new tour can be easily recalculated by $TC_k^{new} = TC_k + c_{i,x} + c_{x,j} - c_{i,j}$. It can be used in all tours so that insertions are substantially accelerated in order to compute the total travel cost, TTC. The best feasible insertion is determined by using the proposed metric in Eqs. (5) and (6) while TTC is determined using the nearest neighbor approach. Note that for insertion of node x to first and last position, the nearest neighbor approach cannot be used but it can be easily modified as given in Tasgetiren, Suganthan, and Pan (2007), (2010). The recreate procedure stops when the last node in U is inserted into all positions in all tours.

To illustrate, we consider the example in Fig. 1. Suppose that $rS_i = 2$. In the ruin phase, two nodes will be removed from the individual λ_i. Suppose we select node 4 from the first tour and node 7

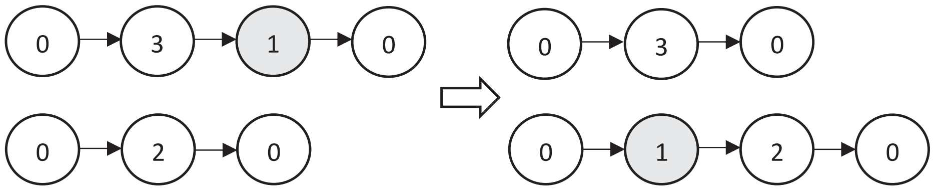

Fig. 2. An example of N_1.

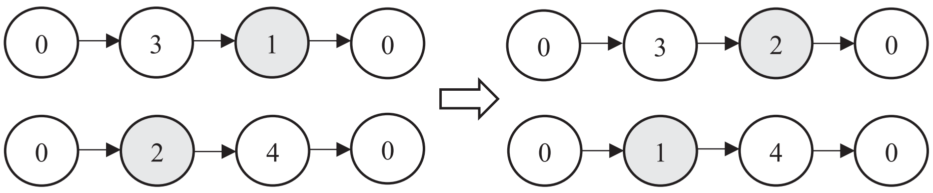

Fig. 4. An example of N_3.

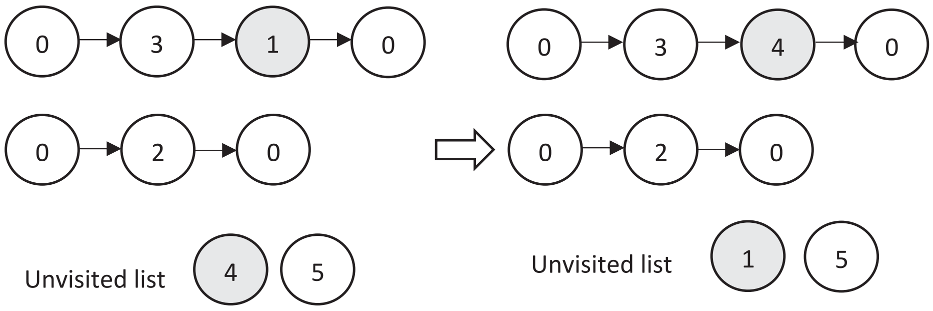

Fig. 5. An example of N_4.

Table 1
Average gaps and running times of constructive heuristics.

	LI1		LI2		LI3		LS1		LS2		VI		KT	
Instance	Gap (%)	t (ms)	Gap (%)	t (ms)	Gap (%)	t (ms)	Gap (%)	t (ms)	Gap (%)	t (ms)	Gap (%)	t (ms)	Gap (%)	t (ms)
C (m = 1)	27	3.49	25	10.86	46	0.07	24	1.12	38	0.88	19	0.68	21	0.78
C (m = 2)	27	4.76	25	30.62	40	0.09	24	2.15	35	0.73	20	2.08	19	2.33
C (m = 3)	25	7.74	21	54.87	36	0.13	22	2.43	32	0.65	16	4.05	16	4.5
C (m = 4)	20	9.27	20	85.67	32	0.16	20	2.27	29	0.58	14	6.32	13	7
S (m = 1)	18	0.84	11	3.73	22	0.04	11	0.74	37	0.52	8	0.45	8	0.53
S (m = 2)	16	1.2	10	9.18	19	0.07	11	0.66	29	0.33	9	1.09	9	1.25
S (m = 3)	13	1.38	7	13.35	15	0.09	10	0.43	23	0.22	7	1.55	7	1.82
S (m = 4)	10	1.47	7	16.68	12	0.1	6	0.35	19	0.17	6	1.84	6	2.23
V (S)	7	1.68	5	23.68	8	0.13	7	0.35	12	0.22	5	3.04	4	3.79
V (C)	7	12.99	8	327.9	12	0.4	12	0.45	15	0.24	5	25.96	6	30.08
Average	17.07	4.48	13.87	57.65	24.19	0.13	14.63	1.09	26.89	0.46	**10.88**	**4.71**	**10.86**	**5.43**

C: Cordeau; S: Solomon; V: Vansteenwegen.

Table 2
ANOVA results for parameters of ES.

	Sum Sq	Df	F value	Pr(> F)
β	0.000033714	3	3.9325	0.01222
μ	0.000003700	3	0.4316	0.73109
β*μ	0.000028355	9	1.1025	0.37397
Residuals	0.000182893	64		

from the second tour. Then, nodes 4 and 7 will be removed from λ_i and added to the end of the unvisited list U as follows:

$T_1 \leftarrow \{2, 8\}$

$T_2 \leftarrow \{1, 5\}$

$U \leftarrow \{3, 6, 9, 10, 4, 7\}$

$TC_1 \leftarrow c_{02} + c_{28} + c_{80}$

$TC_2 \leftarrow c_{01} + c_{15} + c_{50}$

$TTC \leftarrow TC_1 + TC_2$

$f(\mu) \leftarrow P_2 + P_8 + P_1 + P_5$

Then, in the recreate phase, we employ the neighborhood structure, N_0, in order to insert all nodes in U into all positions in T_1 and T_2 in order to find a new feasible offspring with an increased total profit. In other words, first, node 3 is removed from U, and inserted into three positions of both $T_1$and T_2. Suppose that there is no increase in total profit with a feasible insertion from them. Then, node 6 is tried. Suppose that there is no increase in total profit with a feasible insertion from them, again. Then, node 9 is tried. Suppose that there is an increase in total profit with a feasible insertion in position 2 of tour T_2. Next, node 10 is tried. Suppose that there is an increase in total profit with a feasible insertion in position 3 of tour T_1. Now, nodes 4 and 7 are also tried. Suppose that there is no increase in total profit with a feasible insertion from them. Ultimately, we end up with a new offspring with feasible tours and increased profit as follows:

$T_1 \leftarrow \{2, 8, 10\}$

$T_2 \leftarrow \{1, 9, 5\}$

$U \leftarrow \{3, 6, 4, 7\}$

$TC_1 \leftarrow c_{0,2} + c_{2,8} + c_{8,10} + c_{10,0}$

$TC_2 \leftarrow c_{0,1} + c_{1,9} + c_{9,5} + c_{5,0}$

$TTC \leftarrow TC_1 + TC_2$

$f(\mu) \leftarrow P_2 + P_8 + P_{10} + P_1 + P_9 + P_5$

3.4. Local search

We employ a random local search consisting of four different neighborhood structures. Let N_k be the kth neighborhood structure. The

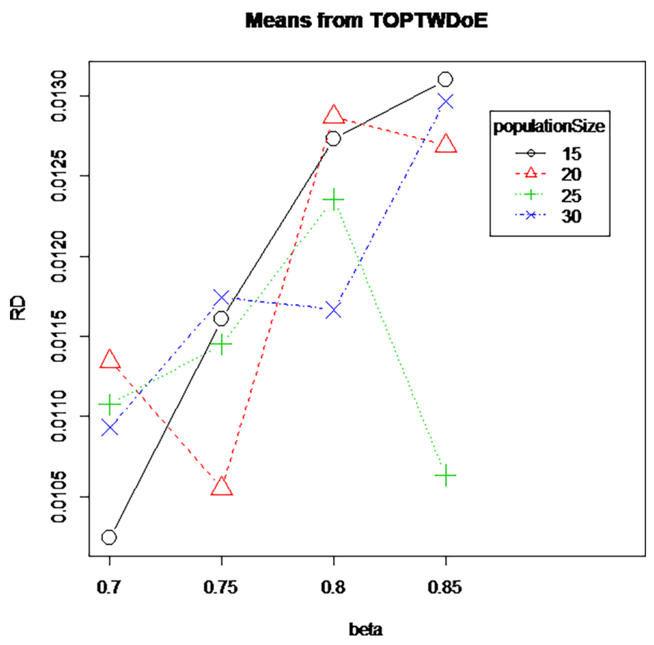

Fig. 6. Plot of β and μ against the average gap/RD.

following four neighborhood structures are proposed for the local search:

- N_1 = *MoveNodeToAnotherTour*: This neighborhood structure chooses a random node from a random tour and moves it to another randomly chosen tour in a random position in that tour. Fig. 2 provides an example where node one in tour one is moved to position one in tour two. Note that this neighborhood is not used when there is only one tour ($m = 1$).
- N_2 = *SwapNodesWithinTour*: This neighborhood structure selects a random tour and then swaps two randomly chosen nodes in that tour. Fig. 3 provides an example where nodes one and three in the tour are swapped.
- N_3 = *SwapNodesBetweenTours*: This neighborhood structure chooses two random tours and two random nodes in those tours and swaps them. Fig. 4 provides an example where node one in tour one is swapped with node two in tour two. Similar to N_1, this neighborhood is not used when there is only one tour ($m = 1$).
- N_4 = *SwapNodeWithUnvisited*: This neighborhood aims at achieving one of the following goals by swapping a visited node with an unvisited node. It either increases profit by swapping a visited node with an unvisited one that has a higher profit, or decreases the total travel cost for the corresponding tour if the profits of the visited and unvisited nodes are the same or both if the swap results in profit increase and travel cost decrease at the same time. Fig. 5 provides an example where node one in tour one is swapped with unvisited node four, making node four visited in tour one and node one unvisited.

Briefly, the local search begins with applying relocation moves, which is applied to each tour separately. They try to relocate nodes within each tour in order to decrease the travel cost for that tour. The backward relocation (BR) procedure takes nodes starting from the last node in the current tour and tries to move the nodes so that they are visited earlier. The forward relocation (FR) procedure starts from the first visited node in the current tour and tries to move the nodes so that they are visited later. Feasibility check and change in travel cost are computed by using the previously mentioned speed-up methods described in Section 2, and the nearest neighbor approach in Section 3.3, respectively. Relocation procedures use first improvement pivoting rule, i.e., the node is relocated as soon as a relocation that lowers the travel cost is found.

If the travel cost is decreased by the relocation moves, it is checked whether one or more unvisited nodes can be added to the current solution by using the neighborhood structure, N_0. Next, a random local search based improvement procedure is executed. One neighborhood

Table 3
Overall comparison of ES with existing algorithms on the instances of Cordeau and Solomon.

	ES		CP	I3CH		GVNS		SSA		ILS		ABC		GRILS		ILS^{I3CH}	SAILS^{I3CH}		ILS^{VNS}	SAILS^{VNS}	
	AG	AT	AG	AG	AT	AG	AT	AG	AT	AG	AT	AG	AT	AG	AT	AG	AG	AT	AG	AG	AT
m = 1																					
C 1–10	0.74	40.7	1.35	1.05	109.0	0.54	12.4	0.97	112.2	4.72	1.8	1.47	78.5	3.73	4.2	0.66	0.24	126.8	0.57	0.66	753.6
C 11–20	3.35	33.7	3.81	4.31	130.2	3.21	24.2	3.75	162.4	9.59	2.0	3.56	103.7	7.99	5.2	2.28	1.83	151.5	1.70	1.84	958.9
S 100	0.02	0.9	0.47	0.69	26.7	1.40	66.5	0.05	22.3	1.94	0.2	0.46	4.4	0.27	1.4	0.00	0.00	31.1	0.00	0.03	78.3
S 200	0.69	35.8	3.55	1.58	132.1	2.15	75.5	1.09	44.7	3.11	1.7	1.56	21.4	2.65	8.4	0.88	0.32	153.7	0.47	0.52	786.4
Avg.	**0.79**	22.9	2.12	1.53	88.6	1.79	57.0	1.03	60.5	3.73	1.2	1.39	33.2	2.59	4.8	**0.70**	**0.39**	103.1	**0.47**	**0.53**	534.6
m = 2																					
C 1–10	1.35	55.5	3.38	1.20	247.1	0.91	39.1	2.54	173.9	6.31	4.8	3.13	149.7	6.85	11.7	2.32	0.87	287.4	1.90	1.43	481.1
C 11–20	2.21	76.1	5.95	3.07	304.6	1.57	82.4	4.24	201.6	8.19	5.2	4.00	221.8	11.61	13.5	2.78	1.83	354.3	2.73	2.38	567.2
S 100	0.22	16.7	1.24	0.50	69.3	0.87	73.9	0.16	34.5	1.96	0.9	0.44	13.0	1.75	4.0	0.12	0.06	80.6	0.01	0.12	63.1
S 200	0.72	80.8	1.44	0.64	463.8	0.99	19.8	1.12	76.9	3.26	2.6	1.32	30.6	3.91	13.7	1.18	0.55	539.5	1.12	0.90	746.0
Avg.	**0.81**	52.4	2.21	0.98	263.8	1.01	51.2	1.35	89.9	3.82	2.6	1.58	64.7	4.48	9.7	1.14	**0.57**	306.9	1.01	**0.87**	427.0
m = 3																					
C 1–10	0.99	95.9	4.70	0.43	424.0	0.43	85.9	2.40	197.0	6.64	9.2	2.97	228.9	7.64	18.4	2.79	1.46	493.2	2.78	2.26	433.8
C 11–20	1.97	123.7	5.67	1.24	497.0	1.15	150.7	3.94	251.8	9.31	9.7	4.36	247.6	11.33	20.3	3.21	1.79	578.1	3.20	2.49	474.5
S 100	0.32	21.5	1.95	0.21	135.9	0.90	91.1	0.45	45.9	2.41	1.5	0.64	22.5	3.20	6.4	0.45	0.25	158.0	0.28	0.63	63.1
S 200	0.10	17.5	0.04	0.02	89.2	0.14	7.3	0.48	52.3	1.13	1.7	0.35	12.5	1.65	14.1	1.65	0.66	103.8	0.33	0.33	295.5
Avg.	**0.55**	43.3	2.12	**0.31**	204.7	**0.60**	68.5	1.18	95.2	3.42	3.7	1.33	75.7	4.30	12.5	1.55	**0.76**	238.1	1.01	0.98	248.6
m = 4																					
C 1–10	1.42	137.7	5.41	0.67	566.5	1.38	127.3	2.53	255.6	7.37	14.1	3.78	233.3	8.37	25.4	3.48	1.98	659.0	3.61	3.30	369.6
C 11–20	2.23	160.3	6.27	0.53	728.6	2.13	232.6	4.03	284.0	8.54	13.7	3.91	267.6	10.41	27.2	3.53	1.89	847.6	3.88	3.24	374.1
S 100	0.55	56.6	2.38	0.26	199.5	1.18	86.6	0.67	58.3	3.30	2.4	1.14	29.5	5.15	8.9	1.11	0.48	232.1	0.85	1.27	61.2
S 200	0.00	0.0	0.00	0.00	0.2	0.00	0.5	0.00	40.4	0.00	1.0	0.00	0.5	0.00	14.3	0.18	0.06	0.2	0.00	0.00	129.4
Avg.	0.69	60.8	2.44	**0.26**	246.6	0.91	80.6	1.12	107.6	3.35	4.9	1.45	77.4	4.44	15.4	1.41	**0.71**	286.9	1.31	1.35	167.2
Grand Avg.	**0.71**	44.9	2.22	**0.77**	200.9	1.08	64.3	1.17	88.3	3.58	3.1	1.44	62.8	3.95	10.6	1.20	**0.61**	233.7	0.95	0.93	344.3

AG = Aggregated gap for instances: AT = Aggregated CPU time for instances.

Table 4
Overall comparison of ES with existing algorithms on the instances of Vansteenwegen.

	ES		CP	I3CH		GVNS		SSA		ILS		ABC		ILS^{100}	ILS^{35}	ILS^{10}	SAILS^{100}	SAILS^{35}	SAILS^{10}
	AG	AT	AG	AG	AT	AG	AT	AG	AT	AG	AT	AG	AT	AG	AG	AG	AG	AG	AG
C 1–10	0.96	78.3	0.89	0.78	326.7	1.25	51.3	1.04	566.0	2.32	30.4	1.05	181.1	0.96	1.11	1.23	0.92	1.05	1.05
S 100	0.15	60.5	0.24	0.03	393.8	1.14	29.7	0.59	90.8	1.80	3.2	0.53	41.2	0.47	0.18	0.19	0.39	0.12	0.12
S 200	0.02	12.8	0.07	0.04	127.2	0.12	3.2	0.08	48.4	0.39	1.5	0.07	5.9	0.02	1.68	1.84	0.09	1.53	1.43
Average	**0.22**	43.7	**0.27**	**0.15**	274.5	0.74	22.1	0.45	145.4	1.30	6.6	0.42	48.0	0.36	0.93	1.02	0.35	0.84	0.80

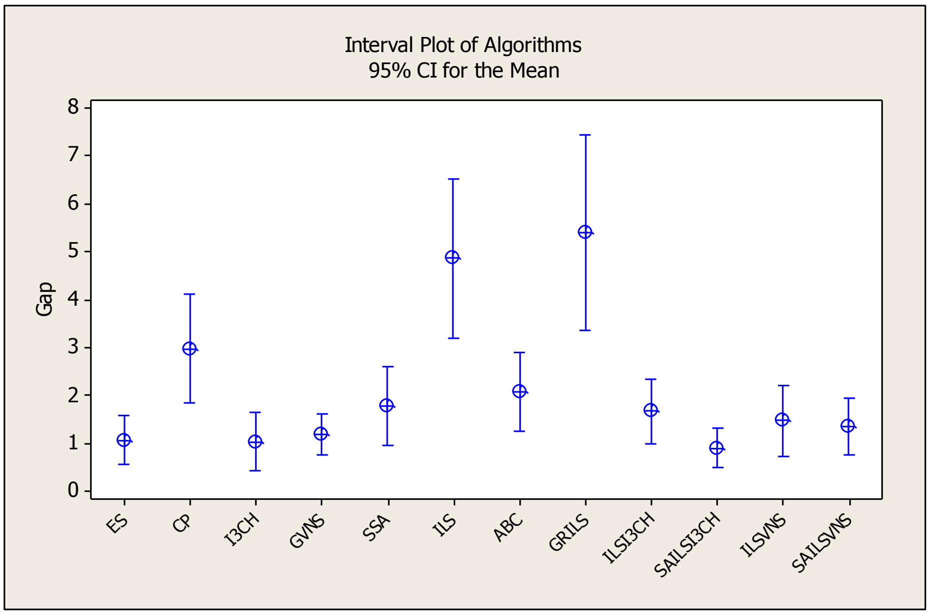

Fig. 7. Interval plot of the mean Gaps for Cordeau and Solomon instances.

Table 5
New best solutions found by ES.

Instance	m	BKS	New BKS
r209	2	1414	1417
pr13	2	832	845
pr15	2	1220	1222
pr10	3	1573	1574
pr03	4	1246	1247
pr08	4	1385	1392
pr20	4	2062	2082

Table 6
CPU Mark results of different algorithms.

Algorithm	CPU	CPU Mark	Multiplier
ES	Intel Core2 Quad Q9400 @ 2.66 GHz	1128	1.72
I3CH	Intel Xeon E5430 @ 2.66 GHz	1133	1.72
GVNS	Intel Pentium 4 3.06 GHz	657	1.00
ILS, SSA	Intel Core2 Duo T9300 @ 2.50 GHz	978	1.49
ABC	AMD Athlon X2 250	991	1.51
ILS^X, SAILS^X	Intel Core i5-650 @ 3.20 GHz	1323	2.01
GRILS	Intel Xeon L5420 @ 2.50 GHz	1046	1.59

amongst four possible neighborhoods is chosen randomly. If the chosen neighborhood results in a decreased travel cost or a better fitness value, then, N_0 is executed again and the iteration counter is reset to one. Otherwise, the iteration counter is incremented by one. The local search continues until a better solution cannot be found in n iterations. The pseudo-code of the local search is given in Algorithm 1.

Algorithm 1. (*LS(g, TTC(g))*).

```
$g' \leftarrow BR(g)$
$g' \leftarrow FR(g')$
**if** $(TTC(g') < TTC(g))$ **then** $g \leftarrow N_0(g')$
$Iter_{max} \leftarrow n$
$Iter \leftarrow 1$
**do while** $(Iter \leq Iter_{max})$
  $g' \leftarrow g$
  $k \leftarrow DU(1, 4)$
  **if** $(k = 1)$ **then** $g' \leftarrow N_1(g')$
  **else if** $(k = 2)$ **then** $g' \leftarrow N_2(g')$
  **else if** $(k = 3)$ **then** $g' \leftarrow N_3(g')$
  **else** $g' \leftarrow N_4(g')$
  **if** $(f(g') > f(g)$ *or* $(TTC(g') < TTC(g))$ **then**
    $g \leftarrow N_0(g')$
    $Iter \leftarrow 1$
  **else**
    $Iter \leftarrow Iter + 1$
  **endif**
**enddo**
**endAlgorithm**
```

3.5. Selection

After having generated offspring population $|\mu|$, they are added to the parent population $|\lambda|$, the combined population $Q = (\lambda \cup \mu)$ will have to be truncated to its initial size before the next iteration. The truncation is done by calculating a goodness score proposed by Lü, Glover, and Hao (2010) and sorting the individuals according to their

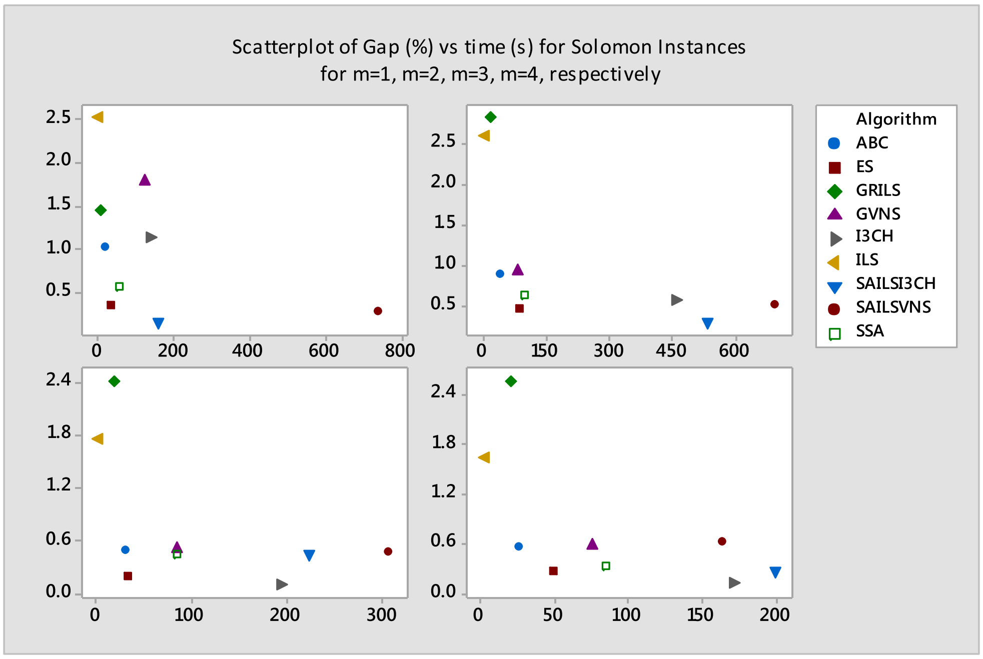

Fig. 8. Scatter plot of the mean gaps vs adjusted CPU times for Solomon instances.

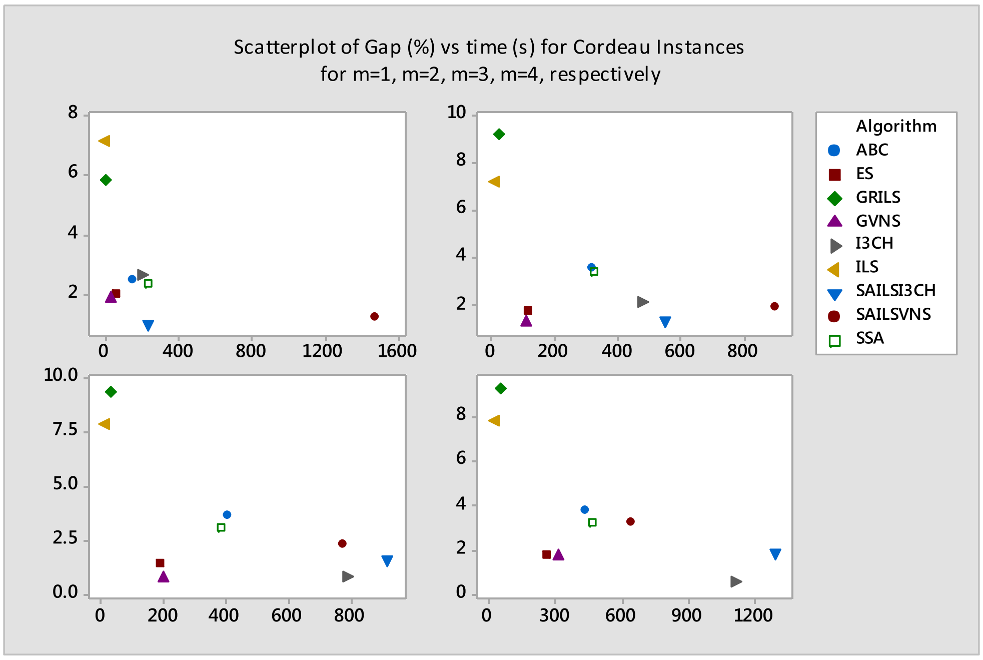

Fig. 9. Scatter plot of the mean gaps vs adjusted CPU times for Cordeau instances.

scores. The individuals with the highest goodness scores are selected for the next generation and the others are removed from the population. The goodness score for an individual is calculated as $GS_i = \beta N(f(Q_i)) + (1-\beta)N(dist_i)$ where β is a weighting parameter, $N(Q_i)$ is a normalization function, $f(Q_i)$ is the fitness value of individual Q_i and $dist_i$ is the distance of individual Q_i to the population. The normalization function is defined as $N(Q_i) = (f(Q_i) - f(Q_{min}))/(f(Q_{max}) - f(Q_{min}) + 1)$, where $f(Q_{min})$ is the minimum and $f(Q_{max})$ is the maximum $f(Q_i)$ value in the population. The distance of individual Q_i to the population is its distance to the most similar individual in the population and is calculated as $dist_i = \min(dist_{ij}, j \neq i)$ where $dist_{ij}$ is the distance between individuals Q_i and Q_j. Consequently, $dist_{ij}$ is the number of different nodes visited in individual Q_i. $dist_{ij}$ can be calculated in shorter time by counting the number of common nodes (*nSize*) in the unvisited list U and subtracting this number from the total number of visited nodes in all routes, instead of comparing all visited nodes in all routes in two individuals and counting the number of common visited nodes: $dist_{ij} = n - nSize$. The pseudo code of the proposed ES is given in Algorithm 2.

Algorithm 2. (*Evolution strategy*).

```
Construct initial population
Apply local search and find $\lambda_{best}$
while (Not TerminationCriterion)
    for $i \leftarrow 1$ to $7 * |\lambda|$
```

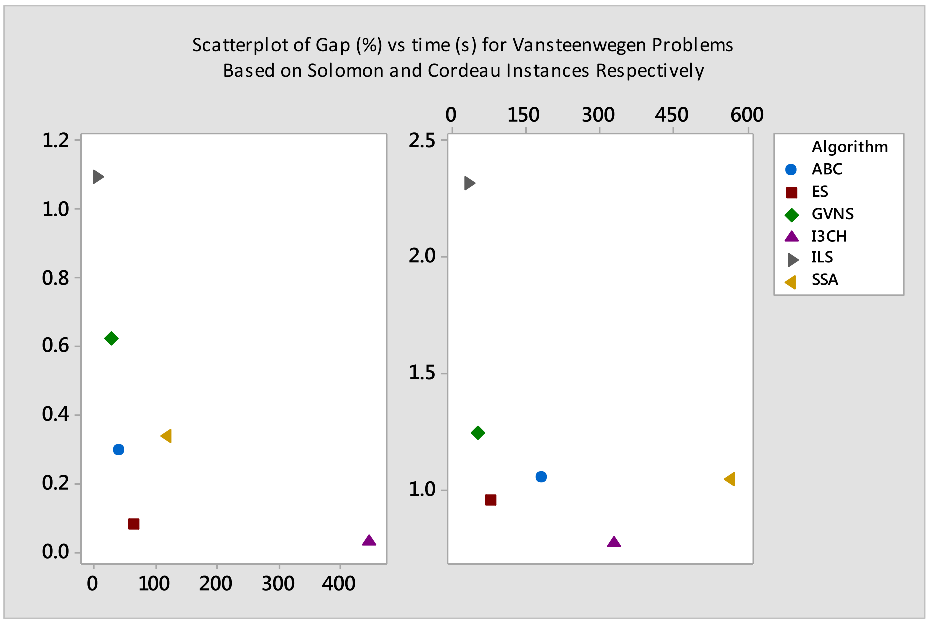

Fig. 10. Scatter plot of the mean gaps vs adjusted CPU times for Vansteenwegen problems.

Table 7
Analysis of Local Search Components.

		ES		No Local Search		Only RL		Only RL + N_0		Only RL + N_1		Only RL + N_2		Only RL + N_3		Only RL + N_4	
		% Gap	time (s)	% Gap	time (s)	% Gap	time (s)	% Gap	time (s)	% Gap	time (s)	% Gap	time (s)	% Gap	time (s)	% Gap	time (s)
m = 1	S	**0.35**	17.7	0.51	22.7	0.53	22.9	0.54	21.2	n/a	n/a	0.53	21.8	n/a	n/a	0.54	19.9
	C	2.04	37.2	2.39	28.8	2.11	34.9	2.11	35.1	n/a	n/a	2.11	40.4	n/a	n/a	2.11	37.8
m = 2	S	0.47	47.6	0.48	43.0	0.47	47.8	0.47	47.7	0.56	51.0	**0.41**	40.9	0.51	45.3	0.52	48.6
	C	**1.78**	65.8	1.86	76.9	1.99	69.7	1.99	69.5	1.89	83.0	1.79	66.4	1.92	87.1	1.81	59.1
m = 3	S	0.21	19.6	0.21	22.1	0.21	26.1	0.21	26.1	0.26	24.3	**0.19**	21.3	0.23	20.1	0.25	19.4
	C	**1.48**	109.8	1.73	105.1	1.84	108.5	1.84	108.3	1.69	99.3	1.59	105.9	1.54	104.3	1.66	120.7
m = 4	S	**0.28**	29.3	0.35	43.0	0.34	26.4	0.34	26.3	0.36	26.1	0.35	32.0	0.33	24.9	0.34	27.1
	C	1.82	149.0	1.99	76.9	1.82	138.8	1.82	136.6	**1.68**	141.4	1.82	137.9	1.75	123.7	1.87	128.1
Opt	S	0.09	37.5	**0.08**	56.2	0.09	71.5	0.22	63.2	**0.08**	55.9	**0.08**	61.4	**0.08**	60.1	0.09	69.2
	C	0.96	78.3	**0.91**	156.3	0.93	127.8	0.19	167.9	0.94	141.3	0.94	115.3	0.99	129.1	0.94	139.1
Avg	S	**0.28**	**30.4**	0.33	37.4	0.33	39.0	0.36	36.9	0.31	39.3	0.31	35.5	0.29	37.6	0.35	36.8
	C	1.62	88.0	1.78	88.8	1.74	95.9	1.59	103.5	**1.55**	116.2	1.65	93.2	**1.55**	111.1	1.68	96.9
Avg		0.95	59.2	1.05	63.1	1.03	67.4	0.97	70.2	0.93	77.8	0.98	64.3	**0.92**	74.3	1.01	66.9
Avg without m = 1																	
	S	**0.26**	33.5	0.28	41.1	0.28	43.0	0.31	40.8	0.31	39.3	**0.26**	38.9	0.29	37.6	0.30	41.1
	C	**1.51**	100.7	1.62	103.8	1.64	111.2	1.46	120.6	1.55	116.2	1.54	106.4	1.55	111.1	1.57	111.7
Avg		**0.89**	**67.1**	0.95	72.5	0.96	77.1	0.89	80.7	0.93	77.8	0.90	72.6	0.92	74.3	0.94	76.4

RL = Backward and forward relocate.

```
        $\mu_i \leftarrow DU(1, |\lambda|)$
        $rS_i \leftarrow rS_i.\ e^{\tau * N(0,1)}$
        **if** $(rS_i < rS_{min})$ **then** $rS_i \leftarrow rS_{min}$
        $rS_{max} \leftarrow$ number of visited cities in $\mu_i$
        **if** $(rS_i > rS_{max})$ **then** $rS_i \leftarrow rS_{max}$
        $\mu_i \leftarrow Ruin(\mu_i)$
        $\mu_i \leftarrow N_o(\mu_i)$
        $\mu_i \leftarrow LS(\mu_i)$
        **if** $(f(\mu_i) > f(\lambda_{best}))$
            $\lambda_{best} \leftarrow \mu_i$
        **else if** $(TTC(\mu_i) < TTC(\lambda_{best})$ **and** $f(\mu_i) = f(\lambda_{best}))$ **then**
            $\lambda_{best} \leftarrow \mu_i$
        **endif**
        Add $\lambda_i$ to population Q
    **endfor**
    Calculate goodness scores (GS) for all individuals in $Q = (\mu \cup \lambda)$
    Sort population Q by GS in a decending order
    Keep $|\lambda|$ best individuals from Q for the next generation and remove others
  **endwhile**
  **return** $\lambda_{best}$
**end Algorithm**
```

4. Computational experiments

The proposed constructive heuristic and evolutionary strategy are coded in C + + and experiments are carried out on a computer having a 2.66 GHz Intel Core2 Quad Q9400 CPU and 3GBs of main memory. Two benchmark sets are used in experiments. There are 144 problem instances in total in these two benchmark sets. The first set is designed by Montemanni and Gambardella (2009) based on their OPTW instances, which in turn are based on Solomon (1987) vehicle routing with time windows test instances and Cordeau, Gendreau, and Laporte (1997) multi-depot vehicle routing problems. These problems have up to four tours. Some of these problem instances have been solved to optimality.

The second benchmark set was designed by Vansteenwegen et al. (2009a, 2009b, 2009c) based on the original Solomon (1987) and Cordeau et al. (1997) instances and use the same number of tours as vehicles, so that, it is possible to visit all nodes. In this way, optimal solutions are equal to the sum of all profits.

4.1. Computational results for constructive heuristics

The KT heuristic proposed in this paper, as well as the heuristics used in Labadie et al. (2011) and Vansteenwegen et al. (2009a, 2009b, 2009c), are also coded and executed on the same machine. In order to compare the performance of the heuristics, the gap is used as a metric which is the difference between the fitness values of the solutions generated by the corresponding heuristic to the best-known or optimum fitness values. The average gap and the average time required to find the solutions by each heuristic for each problem set are given in Table 1.

From Table 1, it can be seen that the proposed KT heuristic and VI heuristic proposed by Vansteenwegen et al. (2009a, 2009b, 2009c) are superior to all the other heuristics since gaps of 10.86 by KT and 10.88 by VI heuristics are smaller than LI1, LI2, LI3, LS1, and LS2 heuristics. The KT heuristic produces slightly better results than the VI heuristic. LI2 performs better than the other four heuristics, however, its running time is very high when compared to the other heuristics.

4.2. Parameter tuning

The proposed ES has two parameters to tune. They are the population size μ and β value used in goodness score calculation. We carried out a full factorial design by using design of experiments (DOE) approach (Montgomery, 2012) and we took the following values for the parameters: $\mu = \{15, 20, 25, 30\}$ and $\beta = \{0.7, 0.75, 0.8, 0.85\}$. We followed the scenario used by Hu and Lim (2014) and Gunawan et al. (2017) to determine the test instances. The chosen instances are c203, c207, pr02, pr07, pr12, pr16, r102, r105, rc107 and rc204 with m equals to four. In the design of ES, there are $4 \times 4 = 16$ algorithm configurations, i.e. treatments. Each instance is run for 5 times for 16 treatments with a maximum CPU time being equal to $T_{max} = 20$ seconds. We calculate the relative deviation (RD/Gap) for each instance-treatment pair as follows:

$$RPD = \left(\frac{F_{best} - \mathrm{F}}{F_{Best}}\right) * 100 \tag{7}$$

where F_i is the total profit generated by the ES in each treatment, and $F_{min}$is the minimum total profit found amongst 16 treatments. For each instance-treatment pair, the average RD value is calculated by taking the average of 10 instances in each treatment. Then, the response variable average (RD) of each treatment is obtained by averaging these RD values of all treatments. ANOVA results are given in Table 2.

Two-way ANOVA results showed that β parameter was significant with a p-value of 0.01222 that is less than $\alpha = 0.05$, while μ and β: μ interaction is not significant with p values of 0.73109 and 0.37397, respectively. However, since β was significant, we examine the interaction plot, which is given in Fig. 6, Fig. 6 indicates that the values of $\mu = 15$ and $\beta = 0.7$ produced the lowest RD/gap.

4.3. Computational results for evolution strategy

The maximum CPU time is limited to 100 s for Solomon instances and 300 s for Cordeau instances. The ES is executed five times on each problem instance. Parameters for the ES are set as follows: $\mu = 15$; $\tau = 1$; $rS_{min} = 2$; and $\beta = 0.7$. It should be noted that the best-known or optimal solutions so far in the literature are carefully updated by considering the best solutions from ES, CP, I3CH, GVNS, SSA, ILS, and ABC as well as from 50 new best known solutions presented very recently in Gunawan et al. (2017), which can be found in Appendix A from Table A1 to A10. We compare the results of the ES to the updated best-known (BK) or optimal (Opt) solutions in the Appendix A. It is important to note that average gaps and CPU times of the ES are used in all comparisons in all tables in Appendix A since ES is a stochastic algorithm. In addition, we also provided the best, average, worst solution values as well as average CPU times.

We compare the performance of the ES with CP, I3CH, GVNS, SSA, ILS, ABC, GRILS, ILS^{I3CH}, SAILS^{I3CH}, ILS^{VNS} and SAILS^{VNS} for Cordeau, Solomon, and Vansteenwegen problems. In addition, we compare the ES with CP, I3CH, GVNS, SSA, ILS, ABC, ILS^{100}, ILS^{35}, ILS^{10}, SAILS^{100}, SAILS^{35} and SAILS^{10} for Vansteenwegen problems. Note that CP is an exact method, whereas ILS and I3CH are deterministic algorithms with only a single run. Only the best solutions values are reported for the SSA algorithm. Finally, the ABC, GRILS, ILS^{I3CH}, SAILS^{I3CH}, ILS^{VNS}, and SAILS^{VNS} are stochastic algorithms. For these reasons, average values of ES, GVNS, ABC, GRILS, and different CPU variants of ILS^{X} and SAILS^{X} are compared to the best values of ILS, I3CH and SSA algorithms.

Even though the detailed results are given in Table A1–A10, in Appendix A for ES, CP, I3CH, GVNS, SSA, ILS, ABC algorithms, the detailed results for GRILS, ILS^{X} and SAILS^{X} algorithms are not provided in the literature. For this reason, we directly take the aggregated values presented in their papers for these three algorithms in the following analyses. Since it is difficult to make an analysis of competing algorithms due to the complexity of computational results, we provide Table 3 and Table 4, where the results are aggregated as in Gunawan et al. (2017). Note that we only evaluate the algorithms in terms of solution quality below.

We analyze the performance of algorithms on instances of Cordeau and Solomon. From Table 3, it can be observed that the best average percent gaps (AG values) for m = 1 are obtained by SAILS^{I3CH}, ILS^{VNS}, SAILS^{VNS} are 0.39%, 0.47%, and 0.53%, respectively. However, ILS^{I3CH} and ES algorithms are very competitive with them having gaps with 0.70% and 0.79%, respectively. For m = 2, the best result with a gap of %0.57 is produced by SAILS^{I3CH} algorithm while ES and SAILS^{VNS} are also very competitive with it by generating gaps with 0.81% and 0.87%, respectively. Regarding m = 3, the best algorithm is I3CH with a gap of 0.31% followed by ES and GVNS with 0.55% and 0.60% gaps, respectively. Finally, for m = 4, again, the best algorithm is I3CH with a gap of 0.26% followed by ES and SAILS^{I3CH} with 0.69% and 0.71% gaps, respectively. However, one can have a look at the grand average of 16 instances. It can be observed from the grand average in Table 3 that the best algorithms are SAILS^{I3CH}, ES and I3CH by generating gaps of 0.61%, 0.71%, and 0.77%, respectively. Note that SAILS^{I3CH} and I3CH algorithms are computationally very expensive ones when compared to ES that will be analyzed by adjusting the CPU times for different machines used by algorithms.

Now, we analyze the performance of algorithms on instances of Vansteenwegen. From Table 4, it can be observed that ILS^{X} and SAILS^{X} algorithms are not so much competitive to I3CH, ES and CP algorithms. The best algorithm is I3CH generating a gap of 0.15% while ES and CP are able to generate very competitive results with gaps of 0.22% and 0.27%, respectively.

Table 3 Overall comparison of ES with existing algorithms on the instances of Cordeau and Solomon

In order to picture out if the differences in Gaps are significant in Table 3, we provide the interval plot of the algorithms in terms of AG (average Gap) values in Fig. 7. Note that, for confidence interval plots, if the confidence interval of algorithm A does not overlap with the confidence interval of algorithm B, differences in mean gaps between A and B are statistically significant and meaningful. As seen in Fig. 7, the differences in Gaps between SAILS^{I3CH}, I3CH, ES and CP, ILS, GRILS are significant; hence, meaningful since their confidence intervals do not overlap. In terms of mean Gap values, the best algorithms are SAILS^{I3CH}, I3CH and ES algorithms. However, SAILS^{I3CH}, ILS^{VNS}; SAILS^{VNS}, ILS^{I3CH}, GVNS, I3CH and ES algorithms are equivalent because their confidence intervals do overlap.

To sum up, ES is very competitive to I3CH and ILS^X and $SAILS^X$ variants in terms of gap values as seen in analyses above. Ultimately, ES is able to find seven new best solutions that are shown in Table 5.

Since the algorithms are tested on different processors, we use https://www.cpubenchmark.net/singleThread.html website in order to obtain the single thread benchmark results to be able to compare the reported CPU times in a fair manner. CPU Mark results for the CPUs used in each study are given in Table 6. We omit CP since it is an exact algorithm with time-limited runs and each instance is run for 30 min.

Since the GVNS has the slowest CPU, we calculate the CPU time result multipliers by calculating the ratio between the GVNS and the other algorithms. We use the multipliers to adjust the average reported CPU times for each algorithm. Scatter plots of mean percent gaps and mean adjusted CPU times are given Fig. 8 for Solomon instances. Note that ILS^{100} and $SAILS^{100}$ are not included in Fig. 8 since their average times (AT values) are not reported in Gunawan et al. (2017).

For m = 1 in Fig. 8, it can be observed that the fastest algorithms are ILS, GRILS, ABC, ES, and SSA while ES weakly dominates them in terms of CPU time. The best algorithm is $SAIL^{SI3CH}$ at the expense of CPU time. $SAIL^{SVNS}$ performed similarly to ES at the expense of extensive CPU time. For m = 2, again, $SAIL^{SI3CH}$ is the best performing algorithm in terms of Gap value at the expense of a very high CPU time. ES dominates both I3CH and $SAIL^{SVNS}$ in terms of both Gap and CPU time. For m = 3, the I3CH is the best performing algorithm in terms of Gap value at the expense of a very high CPU time whereas ES dominates ABC, SSA, GVNS, $SAIL^{SI3CH,}$ and $SAIL^{SVNS}$ in terms of both Gap value and CPU time. For m = 4, the best algorithm is I3CH in terms of Gap value at the expense of a very high CPU time. ES weakly dominates $SAIL^{SI3CH}$ and also dominates SSA, GVNS, and $SAIL^{SVNS}$ in terms of both Gap value and CPU time.

Scatter plots of mean percent gaps and mean adjusted CPU times are given in Fig. 9 for Cordeau instances. For m = 1 in Fig. 9, it can be observed that the best algorithm is $SAIL^{SI3CH}$ in terms of Gap value with a reasonable CPU time. $SAIL^{SVNS}$ is the second one at the expense of a very high CPU time. $SAIL^{SI3CH}$ dominates SSA, ABC, and I3CH in terms of both Gap value and CPU time. ES and GVNS are very competitive with them. For m = 2, GVNS weakly dominates $SAIL^{SI3CH}$ but they are the best performing algorithms in terms of Gap value. GVNS dominates ES, I3CH, ABC, SSA, and $SAIL^{SVNS}$. ES is very competitive to GVNS in terms of both Gap value and CPU time. For m = 3, GVNS and I3CH algorithms are the best ones in terms of Gap values but I3CH is computationally very expensive. ES weakly dominates $SAIL^{SI3CH}$ but dominates SSA, ABC, and $SAIL^{SVNS}$. For m = 4, I3CH is the best performing algorithm in terms of Gap value at the expense of a very high CPU time. ES weakly dominates GVNS and $SAIL^{SI3CH}$ but dominates SSA, ABC, and $SAIL^{SVNS}$. Note that the fastest algorithm is ILS.

Scatter plots of mean percent gaps and mean adjusted CPU times are given in Fig. 10 for Vansteenwegen instances. For Solomon instances in Fig. 10, it can be observed that the best algorithm is I3CH in terms of Gap value with a very high CPU time. ES is the second in terms of Gap value and a reasonable CPU time. ILS is the fastest algorithm again. For Cordeau instances, I3CH is the best algorithm in terms of Gap value at the expense of a very high CPU time. ES is the second one with a very reasonable CPU time. ILS is again the fastest algorithm.

4.4. Analysis of local search components

We carried out experiments in order to analyze the impact of components of local search procedure on the performance of the ES algorithm. The following components are analyzed below: ES without local search, Relocation (RL, both forward and backward) only, Only RL + N_0, Only RL + N_1, Only RL + N_2, Only RL + $N_{3,}$ and Only RL + N_4. We run all eight variants of the ES algorithm on all problem instances. The results are given in Table 7.

It is clear from Table 7 that applying local search improves the solution quality since without the local search, the average gap was 1.05% and it is decreased up to 0.95% with the inclusion of the local search. Only RL + N3 component was the best one with a 0.92% gap but it cannot be applicable to instances with m = 1. In addition, Only RL + N1 was the second best one but again it cannot be applicable to instances with m = 1. Without these two components, the ES performed better than the remaining components by generating an average Gap of 0.95%. In order to see how it performs without m = 1, we provide additional average gap values in Table 7. It can be observed from Table 7 that the ES algorithm has generated 0.89% Gap with an average CPU time of 67.1 s, smaller CPU than all other components.

5. Conclusion

A novel KT constructive heuristic and a self-adaptive evolution strategy for solving the TOPTW are presented in this paper. Three individuals in the initial population are constructed using KT, LS1 and VI heuristics. Then, the rest of the population is constructed by perturbations of these three heuristics in such a way that randomly chosen two neighborhood structures are applied to these three solutions. Offspring are generated by the ruin and recreate algorithm, where ruin size is adaptively updated by the evolution strategy. Four efficient and effective neighborhood structures are designed and used in a random local search scheme. In addition, the speed-up method and the proposed desirability measure are employed in the evolution strategy wherever necessary. For the selection of the population for the next generation, a goodness score is proposed to diversify the population. Extensive computational results are presented and compared to the best performing algorithms from the literature by using interval plots and scatter plots to see differences between algorithms. Computational results show that the evolution strategy is very competitive or even better than the competing algorithms. Ultimately, seven new best-known solutions are presented in the paper and they are given in Appendix B.

Experimental results show that the proposed constructive heuristic can generate good and feasible solutions in a short time. The self-adaptive ES utilizing the ruin and recreate heuristic is a very fast and effective algorithm to solve TOPTW producing high-quality solutions while utilizing less CPU time when compared to the alternative methods. The proposed algorithm achieves very good results even without the local search procedure which shows the effectiveness of the algorithm. Swapping nodes between tours and moving a node to another tour neighborhoods are the better improvement moves among the four moves used in this study when there is more than one tour.

In addition, the proposed self-adaptive ES is simple and can be easily adapted for other combinatorial problems using a problem specific ruin and recreate method. It has only two parameters that also makes adaption to new problems easy.

Acknowledgement

M. Fatih Tasgetiren acknowledges the HUST Project in Wuhan, China. He is partially supported by the National Natural Science Foundation of China with Grant No. 51435009.

Appendix A. Detailed results of the algorithms

See Tables A1–A10.

Table A1
Results for Solomon's problems for m = 1.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg.	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg.	Gap	t (s)	Best	Avg.	Gap	t (s)
c101	320	320	320.0	320	0.0	0.0	320	0.0	320	0.0	21.8	320	320.0	0.0	0.2	320	0.0	20.4	320	0.0	0.4	320	320.0	0.0	0.4	320	320.0	0.0	1.2
c102	360	360	360.0	360	0.0	0.0	360	0.0	360	0.0	28.1	360	360.0	0.0	67.4	360	0.0	20.7	360	0.0	0.3	360	360.0	0.0	0.6	360	360.0	0.0	1.7
c103	400	400	400.0	400	0.0	0.2	390	2.5	400	0.0	27.1	400	396.0	0.0	382.4	400	0.0	24.2	390	2.5	0.5	399	398.0	0.5	3.6	400	397.0	0.8	2.4
c104	420	420	420.0	420	0.0	0.5	400	4.8	420	0.0	27.1	410	410.0	2.4	1005.9	420	0.0	21.9	400	4.8	0.3	420	420.0	0.0	15.2	420	411.0	2.1	2.9
c105	340	340	340.0	340	0.0	0.0	340	0.0	340	0.0	23.4	340	340.0	0.0	3.3	340	0.0	20.6	340	0.0	0.3	340	340.0	0.0	0.5	340	340.0	0.0	1.1
c106	340	340	340.0	340	0.0	0.0	340	0.0	340	0.0	23.6	340	340.0	0.0	6.5	340	0.0	20.3	340	0.0	0.3	340	340.0	0.0	0.5	340	340.0	0.0	1.3
c107	370	370	370.0	370	0.0	0.4	360	2.7	370	0.0	24.7	360	358.0	2.7	0.7	370	0.0	20.5	360	2.7	0.3	370	370.0	0.0	1.3	370	370.0	0.0	1.4
c108	370	370	370.0	370	0.0	0.0	370	0.0	370	0.0	24.8	370	354.0	0.0	1.1	370	0.0	20.5	370	0.0	0.3	370	370.0	0.0	0.9	370	370.0	0.0	1.5
c109	380	380	380.0	380	0.0	0.0	380	0.0	380	0.0	26.3	380	380.0	0.0	30.6	380	0.0	20.5	380	0.0	0.3	380	380.0	0.0	0.5	380	380.0	0.0	1.8
c201	870	860	860.0	860	1.1	20.5	870	0.0	870	0.0	70.1	850	850.0	2.3	0.1	870	0.0	28.3	840	3.4	1.1	870	870.0	0.0	11.2	870	867.0	0.3	3.1
c202	930	930	930.0	930	0.0	14.3	930	0.0	930	0.0	87.6	930	916.0	0.0	78.7	930	0.0	33.5	910	2.2	2.8	930	930.0	0.0	15.3	930	919.0	1.2	5.5
c203	960	960	960.0	960	0.0	0.2	960	0.0	960	0.0	92.3	960	956.0	0.0	329.0	960	0.0	59.6	940	2.1	1.7	960	958.0	0.2	14.8	960	943.0	1.8	8.3
c204	980	970	970.0	970	1.0	0.2	960	2.0	970	1.0	117.4	970	966.0	1.0	974.0	970	1.0	42.3	950	3.1	1.6	971	966.0	1.4	25.7	970	961.0	1.9	11.0
c205	910	900	900.0	900	1.1	0.0	910	0.0	900	1.1	70.7	900	898.0	1.1	12.5	910	0.0	46.9	900	1.1	1.2	904	900.0	1.1	1.7	910	907.0	0.3	4.2
c206	930	930	928.0	920	0.2	81.8	920	1.1	920	1.1	75.7	930	922.0	0.0	34.4	930	0.0	29.0	910	2.2	1.6	930	928.0	0.2	6.2	920	920.0	1.1	5.1
c207	930	930	930.0	930	0.0	30.7	920	1.1	930	0.0	77.4	930	928.0	0.0	36.3	930	0.0	29.1	910	2.2	2.1	920	918.0	1.3	2.3	930	921.0	1.0	5.7
c208	950	950	948.0	940	0.2	17.5	940	1.1	950	0.0	84.0	950	942.0	0.0	74.2	950	0.0	31.2	930	2.1	1.6	950	950.0	0.0	11.8	950	943.0	0.7	6.1
r101	198	198	198.0	198	0.0	3.4	198	0.0	198	0.0	20.4	197	197.0	0.5	0.2	198	0.0	19.7	182	8.1	0.1	191	190.0	4.0	0.3	198	198.0	0.0	0.7
r102	286	286	286.0	286	0.0	0.0	286	0.0	286	0.0	29.3	281	274.8	1.7	13.4	286	0.0	21.0	286	0.0	0.2	282	281.6	1.5	2.2	286	286.0	0.0	1.4
r103	293	293	293.0	293	0.0	0.0	293	0.0	293	0.0	28.8	288	286.0	1.7	33.8	293	0.0	20.3	286	2.4	0.2	292	292.0	0.3	1.1	293	293.0	0.0	1.7
r104	303	303	303.0	303	0.0	5.4	303	0.0	298	1.7	27.3	301	298.6	0.7	72.7	303	0.0	23.2	297	2.0	0.2	301	299.6	1.1	1.7	303	302.6	0.1	2.0
r105	247	247	247.0	247	0.0	0.0	247	0.0	247	0.0	26.0	236	230.6	4.5	0.3	247	0.0	20.3	247	0.0	0.1	245	242.6	1.8	9.1	247	247.0	0.0	0.9
r106	293	293	293.0	293	0.0	0.0	293	0.0	293	0.0	29.4	283	280.4	3.4	19.1	293	0.0	22.1	293	0.0	0.2	290	289.0	1.4	1.6	293	293.0	0.0	1.5
r107	299	297	297.0	297	0.7	0.1	297	0.7	297	0.7	27.8	291	287.2	2.7	50.3	297	0.7	21.5	288	3.7	0.2	299	299.0	0.0	7.1	299	299.0	0.0	1.7
r108	308	308	308.0	308	0.0	0.4	306	0.6	306	0.6	29.7	308	301.4	0.0	50.1	306	0.6	40.6	297	3.6	0.2	308	308.0	0.0	4.6	308	308.0	0.0	2.0
r109	277	277	277.0	277	0.0	0.1	277	0.0	277	0.0	31.1	277	276.4	0.0	10.4	277	0.0	20.4	276	0.4	0.2	277	277.0	0.0	1.6	277	277.0	0.0	1.2
r110	284	284	284.0	284	0.0	7.5	284	0.0	284	0.0	33.9	281	279.2	1.1	16.8	284	0.0	20.8	281	1.1	0.3	284	282.2	0.6	8.1	284	281.4	0.9	1.3
r111	297	297	297.0	297	0.0	0.2	297	0.0	295	0.7	27.7	292	290.6	1.7	54.2	297	0.0	27.9	295	0.7	0.2	297	294.0	1.0	1.7	297	296.4	0.2	1.7
r112	298	298	298.0	298	0.0	3.5	291	2.3	289	3.0	32.0	290	289.6	2.7	31.9	298	0.0	22.3	295	1.0	0.2	298	297.0	0.3	2.4	298	293.7	1.4	1.5
r201	797	795	795.0	795	0.3	6.1	792	0.6	789	1.0	101.8	785	775.6	1.5	6.7	794	0.4	51.1	788	1.1	1.2	788	787.0	1.3	20.3	796	786.2	1.4	4.6
r202	930	923	922.0	921	0.9	88.7	872	6.2	930	0.0	175.6	890	881.4	4.3	13.4	914	1.7	46.4	880	5.4	1.4	904	895.8	3.7	40.6	911	894.6	3.8	7.1
r203	1028	1025	1015.8	1009	1.2	115.0	962	6.4	1020	0.8	221.4	1002	992.2	2.5	34.7	997	3.0	44.2	980	4.7	1.6	1016	1009.0	1.8	25.6	1005	989.6	3.7	11.4
r204	1093	1090	1081.8	1069	1.0	26.1	1048	4.1	1073	1.8	236.9	1077	1074.0	1.5	77.6	1058	3.2	39.7	1073	1.8	1.7	1073	1070.4	2.1	34.0	1075	1068.7	2.2	17.5
r205	953	953	953.0	953	0.0	13.5	892	6.4	946	0.7	129.3	919	905.8	3.6	15.3	946	0.7	37.8	931	2.3	1.4	953	953.0	0.0	32.9	942	931.2	2.3	7.5
r206	1032	1029	1029.0	1029	0.3	43.3	953	7.7	1021	1.1	169.3	982	966.6	4.8	34.9	1020	1.2	40.9	996	3.5	1.5	1021	1018.0	1.4	21.3	1014	1005.1	2.6	9.6
r207	1077	1077	1077.0	1077	0.0	1.7	1012	6.0	1050	2.5	192.8	1027	1022.0	4.6	46.6	1069	0.7	52.5	1038	3.6	2.0	1069	1060.8	1.5	20.9	1058	1042.7	3.2	13.5
r208	1118	1097	1095.4	1094	2.0	55.1	1078	3.6	1092	2.3	230.0	1086	1084.0	2.9	52.9	1079	3.5	35.8	1069	4.4	1.6	1094	1084.6	3.0	26.7	1088	1084.0	3.0	19.8
r209	961	958	949.6	939	1.2	87.4	926	3.6	948	1.4	136.5	933	926.0	2.9	37.3	945	1.7	61.6	926	3.6	2.4	944	934.6	2.7	26.3	924	914.4	4.8	9.2
r210	1000	991	983.0	974	1.7	69.7	939	6.1	982	1.8	176.9	975	961.4	2.5	30.1	973	2.7	53.6	958	4.2	1.9	966	965.4	3.5	29.3	965	955.1	4.5	9.8
r211	1051	1049	1048.4	1046	0.2	4.8	991	5.7	1013	3.6	167.4	1038	1026.0	1.2	22.5	1041	1.0	40.5	1023	2.7	1.6	1023	1019.0	3.0	15.0	1021	1008.5	4.0	11.3
rc101	219	219	219.0	219	0.0	0.1	219	0.0	219	0.0	21.8	219	219.0	0.0	2.1	219	0.0	19.8	219	0.0	0.2	219	219.0	0.0	0.5	219	219.0	0.0	0.7
rc102	266	266	266.0	266	0.0	0.1	266	0.0	266	0.0	25.5	253	246.0	4.9	5.8	266	0.0	20.2	259	2.6	0.2	266	266.0	0.0	0.6	266	266.0	0.0	1.0
rc103	266	266	266.0	266	0.0	0.1	266	0.0	266	0.0	27.1	256	253.2	3.8	23.3	266	0.0	20.7	265	0.4	0.3	266	266.0	0.0	10.7	266	266.0	0.0	1.2
rc104	301	301	301.0	301	0.0	0.1	301	0.0	301	0.0	27.2	301	301.0	0.0	16.2	301	0.0	27.5	297	1.3	0.3	301	301.0	0.0	20.3	301	300.6	0.1	1.4

(*continued on next page*)

Table A1 (*continued*)

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg.	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg.	Gap	t (s)	Best	Avg.	Gap	t (s)
rc105	244	244	244.0	244	0.0	3.2	244	0.0	244	0.0	26.4	231	227.0	5.3	2.7	244	0.0	20.3	221	9.4	0.2	244	244.0	0.0	1.3	244	244.0	0.0	0.9
rc106	252	252	252.0	252	0.0	0.5	252	0.0	250	0.8	25.0	252	252.0	0.0	2.2	252	0.0	21.0	239	5.2	0.2	252	251.4	0.2	8.1	252	249.3	1.1	0.9
rc107	277	277	277.0	277	0.0	0.0	277	0.0	274	1.1	26.3	274	261.2	1.1	15.7	277	0.0	22.0	274	1.1	0.2	277	276.2	0.3	10.4	277	276.4	0.2	1.1
rc108	298	298	298.0	298	0.0	0.0	298	0.0	264	11.4	25.1	298	288.8	0.0	10.4	298	0.0	26.0	288	3.4	0.2	298	298.0	0.0	11.3	298	295.5	0.8	1.1
rc201	795	795	795.0	795	0.0	4.8	785	1.3	795	0.0	80.9	788	784.0	0.9	3.7	795	0.0	44.4	780	1.9	1.0	789	784.0	1.4	24.9	784	781.8	1.7	3.5
rc202	938	934	928.2	905	1.0	27.8	905	3.5	924	1.5	129.3	900	890.6	4.1	12.1	930	0.9	46.3	882	6.0	1.3	931	926.8	1.2	27.3	917	909.8	3.0	5.8
rc203	1003	990	987.6	982	1.5	62.3	930	7.3	966	3.7	134.3	969	954.4	3.4	22.5	967	3.6	32.4	960	4.3	2.7	967	962.4	4.0	24.8	966	960.5	4.2	8.8
rc204	1140	1122	1117.8	1113	1.9	162.2	1066	6.5	1093	4.1	167.5	1137	1102.0	0.3	31.9	1140	0.0	46.5	1117	2.0	2.3	1116	1109.2	2.7	23.4	1121	1103.0	3.2	14.6
rc205	859	859	858.4	856	0.1	1.1	812	5.5	847	1.4	99.2	853	843.8	0.7	7.0	854	0.6	52.6	840	2.2	1.0	854	852.3	0.8	27.2	855	847.2	1.4	4.6
rc206	899	895	890.8	876	0.9	24.9	874	2.8	863	4.0	98.4	879	866.6	2.2	11.6	885	1.6	60.2	860	4.3	1.1	895	890.2	1.0	24.9	877	866.2	3.6	5.5
rc207	983	983	983.0	983	0.0	2.3	952	3.2	957	2.6	122.0	927	911.4	5.7	14.7	977	0.6	68.4	926	5.8	1.3	977	977.0	0.6	15.8	960	930.8	5.3	7.0
rc208	1057	1057	1051.4	1050	0.5	5.8	1012	4.3	1003	5.1	123.0	1014	1000.0	4.1	24.6	1041	1.5	51.2	1037	1.9	2.3	1042	1032.8	2.3	26.3	1024	1004.0	5.0	7.9
					0.3	17.7		2.0		1.1	77.5			1.8	70.9		0.6	33.1		2.5	0.9			1.0	12.6			1.4	4.8

Table A2
Results for Solomon's problems for m = 2.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg.	Gap	t (s)	Best	Gap	t (s)	Best	Gap	T(s)	Best	Avg.	Gap	t (s)	Best	Avg.	Gap	t (s)
c101	590	590	590.0	590	0.0	0.2	590	0.0	590	0.0	53.4	590	590.0	0.0	9.3	590	0.0	26.2	590	0.0	1.4	590	590.0	0.0	12.3	590	589.0	0.2	3.2
c102	660	660	660.0	660	0.0	135.2	660	0.0	660	0.0	78.4	650	648.0	1.5	47.5	660	0.0	26.5	650	1.5	0.9	660	660.0	0.0	12.4	650	649.0	1.7	4.7
c103	720	720	718.0	710	0.3	37.5	720	0.0	720	0.0	116.1	710	704.0	1.4	271.6	720	0.0	26.8	700	2.8	1.2	720	716.0	0.6	19.6	710	693.0	3.8	6.0
c104	760	760	760.0	760	0.0	0.3	760	0.0	760	0.0	94.4	750	746.0	1.3	616.4	760	0.0	28.0	750	1.3	1.5	760	758.0	0.3	13.5	730	729.0	4.1	7.4
c105	640	640	640.0	640	0.0	0.0	640	0.0	640	0.0	70.6	640	640.0	0.0	35.0	640	0.0	26.1	640	0.0	0.8	640	640.0	0.0	2.1	640	640.0	0.0	3.6
c106	620	620	620.0	620	0.0	0.0	620	0.0	620	0.0	149.2	620	620.0	0.0	44.4	620	0.0	25.6	620	0.0	0.8	620	620.0	0.0	2.3	620	620.0	0.0	3.9
c107	670	670	670.0	670	0.0	0.0	670	0.0	670	0.0	65.4	670	670.0	0.0	34.8	670	0.0	26.3	670	0.0	1.4	670	670.0	0.0	2.1	670	670.0	0.0	4.2
c108	680	680	680.0	680	0.0	0.1	680	0.0	680	0.0	85.9	680	680.0	0.0	65.7	680	0.0	26.2	670	1.5	0.8	680	680.0	0.0	3.1	680	677.0	0.4	4.5
c109	720	720	720.0	720	0.0	0.2	720	0.0	720	0.0	69.4	720	716.0	0.0	131.1	720	0.0	26.1	710	1.4	0.9	720	720.0	0.0	8.9	710	705.0	2.1	5.2
c201	1460	1450	1450.0	1450	0.7	2.6	1460	0.0	1450	0.7	321.2	1450	1450.0	0.7	5.8	1450	0.7	64.3	1400	4.1	2.7	1460	1450.0	0.7	22.4	1430	1421.0	2.7	8.7
c202	1470	1470	1460.0	1450	0.7	67.6	1470	0.0	1470	0.0	405.9	1470	1464.0	0.0	21.1	1460	0.7	50.2	1430	2.7	5.1	1460	1458.0	0.8	32.2	1420	1415.0	3.7	11.8
c203	1480	1470	1466.0	1460	0.9	12.5	1470	0.7	1470	0.7	458.2	1470	1464.0	0.7	46.1	1450	2.0	39.4	1430	3.4	3.8	1457	1450.0	2.0	15.9	1430	1420.0	4.1	15.5
c204	1490	1480	1478.0	1470	0.8	47.2	1480	0.7	1480	0.7	498.5	1480	1472.0	0.7	135.5	1450	2.7	38.9	1460	2.0	4.2	1460	1450.0	2.7	17.5	1440	1437.0	3.6	20.6
c205	1470	1470	1470.0	1470	0.0	93.1	1470	0.0	1450	1.4	322.2	1470	1466.0	0.0	10.8	1470	0.0	68.0	1450	1.4	3.1	1470	1470.0	0.0	24.7	1440	1435.0	2.4	10.3
c206	1480	1480	1480.0	1480	0.0	12.5	1480	0.0	1480	0.0	355.9	1480	1472.0	0.0	14.1	1460	1.4	42.4	1440	2.7	2.8	1470	1470.0	0.7	3.8	1440	1437.0	2.9	11.8
c207	1490	1480	1480.0	1480	0.7	29.8	1480	0.7	1470	1.3	423.5	1490	1484.0	0.0	18.1	1480	0.7	87.7	1450	2.7	3.2	1480	1477.0	0.9	28.0	1450	1441.0	3.3	12.2
c208	1490	1480	1480.0	1480	0.7	2.5	1480	0.7	1470	1.3	424.3	1490	1480.0	0.0	18.8	1460	2.0	38.4	1460	2.0	2.8	1479	1472.0	1.2	9.2	1450	1446.0	3.0	13.2
r101	349	349	349.0	349	0.0	0.1	349	0.0	349	0.0	42.1	349	349.0	0.0	4.0	349	0.0	25.0	330	5.4	0.4	345	342.8	1.8	2.9	349	349.0	0.0	2.2
r102	508	508	508.0	508	0.0	0.3	470	7.5	508	0.0	62.4	499	492.8	1.8	17.4	508	0.0	43.9	508	0.0	0.9	508	508.0	0.0	26.1	508	504.8	0.6	3.6
r103	522	520	520.0	520	0.4	2.1	515	1.3	519	0.6	68.3	508	506.6	2.7	37.4	519	0.6	32.3	513	1.7	0.9	519	517.2	0.9	19.1	517	516.0	1.1	4.4
r104	552	549	547.2	546	0.9	3.3	537	2.7	549	0.5	75.3	541	539.2	2.0	93.7	548	0.7	29.1	539	2.4	1.5	549	549.0	0.5	17.3	541	538.6	2.4	5.1
r105	453	447	447.0	447	1.3	2.4	453	0.0	447	1.3	56.2	445	442.6	1.8	5.3	453	0.0	25.4	430	5.1	0.8	450	448.0	1.1	20.7	453	453.0	0.0	2.9
r106	529	529	529.0	529	0.0	0.2	529	0.0	529	0.0	63.5	529	524.4	0.0	31.2	529	0.0	27.2	529	0.0	0.9	529	529.0	0.0	11.1	525	522.5	1.2	3.8
r107	538	538	538.0	538	0.0	0.2	537	0.2	533	0.9	63.2	527	522.2	2.0	56.5	532	1.1	37.0	529	1.7	1.0	520	518.8	3.6	21.9	535	529.9	1.5	4.6
r108	560	560	558.8	558	0.2	39.0	558	0.4	550	1.8	65.7	551	545.6	1.6	179.5	558	0.4	67.5	549	2.0	1.4	557	556.0	0.7	13.7	558	550.9	1.6	5.2
r109	506	506	506.0	506	0.0	2.1	505	0.2	506	0.0	60.5	506	501.6	0.0	19.7	506	0.0	45.5	498	1.6	0.5	504	502.8	0.6	8.9	498	498.0	1.6	3.6
r110	525	525	525.0	525	0.0	2.4	500	4.8	525	0.0	68.9	523	523.0	0.4	65.5	525	0.0	27.5	515	1.9	1.0	525	525.0	0.0	12.4	508	501.9	4.4	3.9
r111	544	541	541.0	541	0.6	4.3	520	4.4	542	0.4	67.0	533	530.6	2.0	115.1	544	0.0	54.0	535	1.7	0.6	544	544.0	0.0	11.8	539	534.8	1.7	4.4
r112	544	544	542.2	535	0.3	14.8	526	3.3	534	1.8	63.0	542	536.6	0.4	98.8	542	0.4	25.1	515	5.3	0.5	544	544.0	0.0	11.5	536	523.1	3.8	4.1
r201	1256	1254	1247.2	1241	0.7	107.2	1246	0.8	1254	0.2	333.6	1235	1225.8	1.7	15.0	1242	1.1	73.7	1231	2.0	2.1	1239	1231.2	2.0	23.0	1236	1212.3	3.5	9.6
r202	1348	1347	1341.6	1336	0.5	103.3	1348	0.0	1344	0.3	588.5	1327	1306.2	1.6	15.4	1334	1.0	50.4	1270	5.8	2.3	1344	1341.6	0.5	40.2	1298	1289.5	4.3	11.0
r203	1418	1416	1413.8	1408	0.3	55.7	1413	0.4	1416	0.1	815.9	1407	1392.6	0.8	17.7	1414	0.3	196.2	1377	2.9	1.9	1414	1410.0	0.6	43.7	1383	1372.0	3.2	14.7
r204	1458	1458	1458.0	1458	0.0	43.3	1452	0.4	1458	0.0	33.5	1458	1452.6	0.0	8.9	1447	0.8	70.9	1440	1.2	3.4	1457	1453.4	0.3	35.6	1440	1433.1	1.7	18.9
r205	1386	1383	1380.2	1379	0.4	138.7	1371	1.1	1380	0.4	606.1	1362	1351.6	1.7	20.1	1363	1.7	51.5	1338	3.5	2.8	1375	1365.8	1.5	47.1	1344	1328.6	4.1	13.3
r206	1450	1440	1438.2	1436	0.8	128.9	1433	1.2	1427	1.6	739.9	1435	1429.8	1.0	17.1	1430	1.4	113.7	1401	3.4	2.8	1430	1426.8	1.6	48.4	1398	1388.9	4.2	14.3
r207	1458	1458	1455.4	1453	0.2	89.6	1443	1.0	1458	0.0	453.8	1454	1449.8	0.3	12.4	1452	0.4	158.8	1428	2.1	1.7	1455	1448.4	0.7	35.7	1431	1422.9	2.4	17.7
r208	1458	1458	1458.0	1458	0.0	0.4	1458	0.0	1458	0.0	4.3	1458	1458.0	0.0	8.3	1457	0.1	56.0	1458	0.0	1.6	1458	1457.8	0.0	12.6	1458	1455.4	0.2	20.2
r209	1414	**1417**	1404.0	1394	0.7	157.3	1391	1.6	1404	0.7	600.3	1402	1389.6	0.8	16.3	1404	0.7	89.5	1345	4.9	2.6	1400	1390.2	1.7	35.0	1360	1344.3	4.9	14.4
r210	1427	1420	1415.4	1411	0.8	107.4	1407	1.4	1415	0.8	728.5	1404	1391.0	1.6	16.0	1414	0.9	77.8	1365	4.3	1.9	1419	1414.2	0.9	38.2	1371	1365.2	4.3	14.1
r211	1458	1458	1457.0	1455	0.1	94.6	1451	0.5	1450	0.5	890.9	1455	1452.6	0.2	14.8	1451	0.5	66.9	1422	2.5	1.9	1457	1451.2	0.5	29.3	1413	1402.3	3.8	16.5
rc101	427	427	427.0	427	0.0	0.1	427	0.0	427	0.0	52.3	427	420.6	0.0	2.8	427	0.0	25.3	427	0.0	0.6	427	425.8	0.3	5.4	427	427.0	0.0	2.5
rc102	505	505	503.4	497	0.3	63.7	504	0.2	505	0.0	59.8	504	485.4	0.2	8.4	505	0.0	42.4	494	2.2	0.8	505	505.0	0.0	5.1	504	501.8	0.6	3.2
rc103	524	523	523.0	523	0.2	1.0	505	3.6	519	1.0	60.3	514	502.0	1.9	26.8	523	0.2	26.5	519	1.0	1.1	524	523.0	0.2	8.7	523	515.1	1.7	3.5
rc104	575	574	573.0	569	0.3	19.5	575	0.0	556	3.3	59.9	562	554.8	2.3	43.6	575	0.0	63.1	565	1.7	0.7	575	572.8	0.4	26.2	564	555.0	3.5	3.9

(*continued on next page*)

Table A2 (*continued*)

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg.	Gap	t (s)	Best	Gap	t (s)	Best	Gap	T(s)	Best	Avg.	Gap	t (s)	Best	Avg.	Gap	t (s)
rc105	480	480	480.0	480	0.0	78.8	465	3.1	480	0.0	58.6	478	456.8	0.4	7.1	480	0.0	56.3	459	4.4	0.8	476	475.2	1.0	28.2	478	476.2	0.8	2.9
rc106	483	483	483.0	483	0.0	64.5	480	0.6	481	0.4	56.0	483	480.4	0.0	11.0	483	0.0	47.3	458	5.2	0.6	483	483.0	0.0	12.6	481	472.3	2.2	2.9
rc107	534	529	529.0	529	0.9	7.3	524	1.9	529	0.9	62.9	529	524.0	0.9	26.0	529	0.9	26.8	515	3.6	0.5	534	531.4	0.5	19.8	520	512.0	4.1	3.2
rc108	556	553	553.0	553	0.5	5.9	546	1.8	547	1.6	61.4	553	544.6	0.5	36.8	554	0.4	36.1	546	1.8	0.6	556	554.0	0.4	17.3	537	525.2	5.5	3.3
rc201	1385	1375	1369.6	1355	1.1	84.1	1310	5.4	1384	0.1	267.4	1369	1359.6	1.2	7.3	1377	0.6	73.3	1305	5.8	1.9	1377	1373.2	0.9	18.5	1345	1327.4	4.2	8.7
rc202	1512	1511	1503.6	1492	0.6	65.3	1494	1.2	1500	0.8	386.7	1476	1465.0	2.4	16.8	1499	0.9	51.8	1461	3.4	2.1	1475	1472.6	2.6	36.8	1480	1441.0	4.7	11.0
rc203	1632	1629	1605.8	1589	1.6	101.1	1534	6.0	1627	0.3	482.8	1570	1561.6	3.8	11.8	1576	3.4	94.9	1573	3.6	2.0	1594	1593.6	2.4	42.0	1560	1535.6	5.9	14.6
rc204	1716	1706	1701.6	1697	0.8	115.6	1660	3.3	1704	0.7	760.8	1713	1699.2	0.2	8.8	1674	2.4	115.6	1656	3.5	2.1	1688	1682.0	2.0	49.6	1656	1642.0	4.3	18.5
rc205	1458	1458	1427.4	1388	2.1	105.8	1458	0.0	1452	0.4	311.0	1407	1396.0	3.5	8.5	1458	0.0	50.5	1381	5.3	3.2	1449	1446.0	0.8	26.2	1401	1383.8	5.1	10.2
rc206	1552	1538	1521.8	1492	1.9	112.5	1469	5.3	1525	1.7	335.9	1531	1507.4	1.4	18.1	1528	1.5	50.8	1495	3.7	1.9	1506	1503.8	3.1	28.1	1505	1460.1	5.9	11.4
rc207	1599	1587	1569.0	1554	1.9	111.6	1547	3.3	1582	1.1	408.6	1563	1546.6	2.3	18.3	1574	1.6	73.5	1531	4.3	2.7	1567	1561.8	2.3	37.7	1506	1492.4	6.7	12.6
rc208	1692	1691	1683.4	1667	0.5	190.6	1636	3.3	1669	1.4	564.4	1687	1669.6	0.3	12.5	1675	1.0	130.4	1606	5.1	1.7	1653	1647.4	2.6	46.1	1620	1584.6	6.3	15.0
					0.5	47.6		1.3		0.6	259.5			0.9	47.8		0.6	54.9		2.6	1.7			0.9	21.5			2.8	8.7

Table A3
Results for Solomon's problems for m = 3.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
c101	810	810	810.0	810	0.0	1.0	810	0.0	810	0.0	303.1	810	808	0.0	18.0	810	0.0	31.7	790	2.5	1.1	810	810.0	0.0	14.7	810	806	0.5	5.2
c102	920	920	920.0	920	0.0	16.1	920	0.0	920	0.0	237.5	920	916	0.0	94.1	920	0.0	32.7	890	3.3	2.1	920	920.0	0.0	5.8	900	888	3.5	7.0
c103	990	980	980.0	980	1.0	32.2	970	2.0	990	0.0	156.5	970	964	2.0	332.3	980	1.0	37.3	960	3.0	2.2	980	976.0	1.4	16.4	960	942	4.8	9.4
c104	1030	1030	1022.0	1020	0.8	6.0	1020	1.0	1030	0.0	155.9	1020	1012	1.0	604.0	1010	1.9	33.1	1010	1.9	1.3	1014	1008.0	2.1	31.5	990	977	5.1	12.0
c105	870	870	870.0	870	0.0	24.9	860	1.1	870	0.0	110.2	870	864	0.0	36.5	870	0.0	32.9	840	3.4	1.0	870	868.0	0.2	14.3	860	849	2.4	5.9
c106	870	870	870.0	870	0.0	26.6	870	0.0	870	0.0	227.4	870	866	0.0	50.2	870	0.0	39.6	840	3.4	1.1	870	870.0	0.0	16.5	860	853	2.0	6.4
c107	910	910	910.0	910	0.0	3.1	900	1.1	910	0.0	151.8	910	906	0.0	63.4	910	0.0	33.4	900	1.1	1.5	910	910.0	0.0	15.4	890	883	3.0	6.8
c108	920	920	916.0	910	0.4	46.0	920	0.0	920	0.0	212.6	920	914	0.0	105.9	920	0.0	33.1	900	2.2	1.2	920	920.0	0.0	14.0	900	892	3.0	7.2
c109	970	970	964.0	960	0.6	4.2	960	1.0	960	1.0	157.2	960	958	1.0	180.7	970	0.0	43.5	950	2.1	2.0	966	962.0	0.8	18.2	940	925	4.6	8.4
c201	1810	1810	1806.0	1800	0.2	22.2	1810	0.0	1810	0.0	3.8	1810	1800	0.0	3.7	1800	0.6	48.1	1750	3.3	2.2	1810	1808.0	0.1	35.4	1770	1728	4.5	11.5
c202	1810	1810	1808.0	1800	0.1	18.8	1810	0.0	1810	0.0	7.2	1810	1806	0.0	8.8	1790	1.1	68.3	1750	3.3	2.0	1793	1792.0	1.0	39.8	1730	1724	4.8	13.7
c203	1810	1810	1808.0	1800	0.1	37.8	1810	0.0	1810	0.0	7.9	1810	1804	0.0	10.9	1770	2.2	111.9	1760	2.8	2.0	1774	1766.0	2.4	45.3	1740	1726	4.6	16.0
c204	1810	1810	1810.0	1810	0.0	51.7	1810	0.0	1810	0.0	11.2	1810	1802	0.0	10.9	1750	3.3	54.5	1780	1.7	1.5	1762	1758.0	2.9	35.8	1760	1753	3.1	20.0
c205	1810	1810	1810.0	1810	0.0	4.8	1810	0.0	1810	0.0	12.6	1810	1810	0.0	4.9	1810	0.0	42.5	1770	2.2	2.5	1810	1804.0	0.3	43.5	1750	1735	4.1	13.0
c206	1810	1810	1810.0	1810	0.0	2.5	1810	0.0	1810	0.0	18.0	1810	1810	0.0	6.5	1780	1.7	40.8	1770	2.2	1.5	1810	1810.0	0.0	1.3	1770	1740	3.9	14.3
c207	1810	1810	1810.0	1810	0.0	1.9	1810	0.0	1810	0.0	16.9	1810	1810	0.0	7.6	1800	0.6	59.5	1810	0.0	3.4	1810	1810.0	0.0	5.1	1750	1739	3.9	14.8
c208	1810	1810	1810.0	1810	0.0	1.3	1810	0.0	1810	0.0	20.6	1810	1810	0.0	8.3	1800	0.6	52.2	1810	0.0	2.4	1810	1810.0	0.0	2.0	1770	1751	3.3	16.1
r101	484	481	481.0	481	0.6	0.1	484	0.0	481	0.6	67.7	481	477.4	0.6	7.7	484	0.0	48.6	481	0.6	0.8	480	477.2	1.4	11.4	484	482.4	0.3	3.7
r102	694	691	691.0	691	0.4	4.9	662	4.6	691	0.4	111.4	686	674.2	1.2	21.9	694	0.0	43.5	685	1.3	1.0	691	690.6	0.5	19.4	686	676.1	2.6	5.7
r103	747	740	740.0	740	0.9	7.6	716	4.1	740	0.9	125.6	730	724.4	2.3	68.4	736	1.5	46.1	720	3.6	2.0	747	746.0	0.1	39.2	730	716.8	4.0	7.0
r104	778	778	778.0	778	0.0	90.3	754	3.1	777	0.1	128.3	770	762	1.0	103.7	777	0.1	71.2	765	1.7	1.5	777	777.0	0.1	25.7	761	752.4	3.3	8.2
r105	620	620	619.2	616	0.1	1.8	615	0.8	619	0.2	165.8	615	612.4	0.8	11.0	619	0.2	36.5	609	1.8	2.3	620	619.3	0.1	24.9	619	614.4	0.9	4.7
r106	729	729	728.0	724	0.1	9.3	716	1.8	729	0.0	122.7	723	713.4	0.8	47.0	729	0.0	38.6	719	1.4	2.1	726	725.8	0.4	24.5	715	710.1	2.6	6.2
r107	760	759	759.0	759	0.1	38.7	739	2.8	759	0.1	120.5	759	755	0.1	110.1	759	0.1	90.8	747	1.7	1.1	760	760.0	0.0	24.2	745	735	3.3	7.3
r108	797	797	797.0	797	0.0	2.8	771	3.3	797	0.0	135.7	781	770.4	2.0	187.9	789	1.0	81.8	790	0.9	3.1	792	785.8	1.4	27.0	778	771.2	3.2	8.2
r109	710	710	706.4	701	0.5	5.5	691	2.7	710	0.0	103.6	700	699.6	1.4	24.1	702	1.1	40.2	699	1.5	1.8	710	704.0	0.8	25.9	702	697.4	1.8	5.8
r110	737	736	736.0	736	0.1	9.9	720	2.3	736	0.1	109.9	734	716.2	0.4	69.8	734	0.4	36.8	711	3.5	1.4	737	734.0	0.4	29.7	704	686.8	6.8	6.2
r111	774	774	773.2	773	0.1	45.6	756	2.3	773	0.1	118.1	761	749.4	1.7	62.6	771	0.4	46.9	764	1.3	1.8	773	771.0	0.4	20.8	755	746.1	3.6	6.9
r112	776	776	775.6	774	0.1	72.8	749	3.5	776	0.0	110.5	757	751	2.4	173.0	776	0.0	91.8	758	2.3	1.1	772	770.8	0.7	28.2	744	731.2	5.8	6.4
r201	1442	1440	1435.8	1432	0.4	58.4	1440	0.1	1439	0.2	981.2	1423	1416.4	1.3	6.4	1429	0.9	50.3	1408	2.4	2.4	1441	1434.8	0.5	39.7	1420	1409.6	2.2	11.2
r202	1458	1458	1457.4	1455	0.0	21.6	1458	0.0	1458	0.0	15.3	1458	1450.2	0.0	21.0	1458	0.0	59.0	1443	1.0	2.7	1458	1458.0	0.0	5.8	1458	1449.1	0.6	11.6
r203	1458	1458	1458.0	1458	0.0	0.3	1458	0.0	1458	0.0	0.2	1458	1458	0.0	9.0	1458	0.0	39.9	1458	0.0	1.6	1458	1458.0	0.0	0.8	1458	1458	0.0	13.2
r204	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	11.9	1458	0.0	38.4	1458	0.0	1.0	1458	1458.0	0.0	0.4	1458	1458	0.0	14.7
r205	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.3	1458	1458	0.0	9.0	1458	0.0	39.3	1458	0.0	1.1	1458	1458.0	0.0	1.3	1458	1458	0.0	13.2
r206	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.3	1458	1458	0.0	9.0	1458	0.0	38.9	1458	0.0	1.1	1458	1458.0	0.0	0.6	1458	1458	0.0	13.6
r207	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.3	1458	1458	0.0	0.2	1458	0.0	38.5	1458	0.0	1.0	1458	1458.0	0.0	0.4	1458	1458	0.0	15.0
r208	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.1	1458	0.0	38.4	1458	0.0	0.8	1458	1458.0	0.0	0.2	1458	1458	0.0	16.1
r209	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.3	1458	1458	0.0	0.2	1458	0.0	40.1	1458	0.0	1.1	1458	1458.0	0.0	0.9	1458	1458	0.0	14.3
r210	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	9.7	1458	0.0	40.0	1458	0.0	1.2	1458	1458.0	0.0	0.6	1458	1458	0.0	13.6
r211	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.4	1458	1458	0.0	0.2	1458	0.0	38.6	1458	0.0	1.0	1458	1458.0	0.0	0.4	1458	1458	0.0	15.7
rc101	621	621	616.8	614	0.7	5.9	621	0.0	621	0.0	87.6	614	602.4	1.1	5.7	621	0.0	34.2	604	2.7	1.4	621	617.4	0.6	23.5	616	613	1.3	4.2
rc102	714	710	710.0	710	0.6	10.0	703	1.5	714	0.0	105.4	705	692	1.3	14.0	710	0.6	41.3	698	2.2	1.3	713	710.3	0.5	26.3	710	703	1.5	4.9
rc103	764	764	757.0	747	0.9	13.7	702	8.1	764	0.0	105.7	764	741.2	0.0	47.0	764	0.0	44.6	747	2.2	1.1	754	748.6	2.0	16.5	742	733.1	4.0	5.7
rc104	835	834	833.2	832	0.2	29.8	835	0.0	834	0.1	124.4	834	826	0.1	85.6	814	2.5	33.0	822	1.6	1.3	831	828.0	0.8	25.9	827	807.4	3.3	6.2

(*continued on next page*)

Table A3 (*continued*)

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
rc105	682	682	682.0	682	0.0	2.4	666	2.3	682	0.0	90.4	680	670.2	0.3	20.8	682	0.0	32.0	654	4.1	0.8	682	682.0	0.0	28.7	681	670	1.8	4.9
rc106	706	706	705.8	705	0.0	17.8	693	1.8	706	0.0	88.1	695	692.6	1.6	17.4	706	0.0	61.5	678	4.0	1.0	696	696.4	1.4	24.4	687	676.7	4.2	4.6
rc107	773	768	768.0	768	0.6	17.7	759	1.8	762	1.4	98.7	762	756.4	1.4	30.7	773	0.0	43.8	745	3.6	0.9	772	769.8	0.4	34.7	759	739.7	4.3	5.1
rc108	795	795	792.8	789	0.3	77.5	769	3.3	789	0.8	107.4	782	773	1.6	48.2	778	2.1	52.0	757	4.8	1.1	782	781.2	1.7	26.2	763	754.1	5.1	5.3
rc201	1698	1698	1685.8	1668	0.7	94.7	1693	0.3	1693	0.3	575.2	1682	1676	0.9	11.6	1681	1.0	98.3	1625	4.3	1.9	1689	1681.0	1.0	32.1	1657	1638	3.5	11.2
rc202	1724	1724	1720.4	1714	0.2	89.9	1724	0.0	1724	0.0	1.1	1709	1702.8	0.9	10.5	1714	0.6	48.3	1686	2.2	1.7	1724	1717.4	0.4	12.7	1708	1691.4	1.9	12.3
rc203	1724	1724	1724.0	1724	0.0	0.7	1724	0.0	1724	0.0	0.3	1724	1724	0.0	7.9	1724	0.0	39.1	1724	0.0	2.9	1724	1724.0	0.0	2.1	1724	1724	0.0	14.2
rc204	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.3	1724	1724	0.0	0.1	1724	0.0	38.3	1724	0.0	1.0	1724	1724.0	0.0	1.1	1724	1724	0.0	15.3
rc205	1719	1709	1705.4	1691	0.8	64.7	1709	0.6	1719	0.0	734.4	1706	1702.2	0.8	11.6	1709	0.6	122.3	1659	3.5	2.4	1709	1704.8	0.8	22.7	1687	1669.2	2.9	12.2
rc206	1724	1724	1724.0	1724	0.0	0.6	1724	0.0	1724	0.0	0.5	1724	1724	0.0	8.7	1724	0.0	46.0	1708	0.9	1.3	1724	1724.0	0.0	4.3	1724	1713.1	0.6	13.7
rc207	1724	1724	1724.0	1724	0.0	0.5	1724	0.0	1724	0.0	0.3	1724	1724	0.0	8.7	1724	0.0	40.4	1713	0.6	1.5	1724	1724.0	0.0	1.9	1724	1714.8	0.5	14.2
rc208	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.4	1724	1724	0.0	0.2	1724	0.0	39.1	1724	0.0	1.1	1724	1724.0	0.0	1.1	1724	1724	0.0	15.8
					0.2	19.6		1.0		0.1	113.4			0.5	50.7		0.5	49.0		1.8	1.6			0.5	17.7			2.5	10.1

Table A4
Results for Solomon's problems for m = 4.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
c101	1020	1020	1020.0	1020	0.0	53.0	1020	0.0	1020	0.0	176.4	1020	1014	0.0	22.1	1020	0.0	37.1	1000	2.0	3.8	1020	1020.0	0.0	13.6	1020	1005.0	1.5	7.4
c102	1150	1140	1136.0	1130	1.2	51.6	1140	0.9	1150	0.0	274.1	1140	1132	0.9	58.2	1150	0.0	40.0	1090	5.2	1.8	1150	1150.0	0.0	27.3	1090	1083.0	5.8	9.6
c103	1210	1210	1196.0	1180	1.2	49.2	1180	2.5	1210	0.0	301.1	1180	1178	2.5	221.2	1190	1.7	38.1	1150	5.0	2.5	1210	1208.0	0.2	43.5	1150	1130.0	6.6	12.9
c104	1260	1260	1254.0	1250	0.5	53.4	1240	1.6	1260	0.0	281.8	1230	1226	2.4	513.0	1230	2.4	41.0	1220	3.2	3.0	1246	1240.0	1.6	38.1	1200	1187.0	5.8	16.5
c105	1070	1060	1058.0	1050	1.1	18.8	1060	0.9	1060	0.9	333.2	1060	1060	0.9	33.3	1060	0.9	36.8	1030	3.7	1.8	1060	1060.0	0.9	18.6	1040	1031.0	3.6	8.2
c106	1080	1080	1068.0	1060	1.1	63.2	1070	0.9	1080	0.0	179.4	1060	1052	1.9	51.5	1080	0.0	69.0	1040	3.7	2.1	1080	1080.0	0.0	31.7	1040	1038.0	3.9	8.9
c107	1120	1110	1110.0	1110	0.9	16.8	1110	0.9	1120	0.0	209.3	1120	1112	0.0	62.0	1120	0.0	45.5	1100	1.8	2.0	1120	1120.0	0.0	23.4	1080	1071.0	4.4	9.5
c108	1140	1130	1126.0	1120	1.2	98.8	1120	1.8	1130	0.9	373.1	1130	1116	0.9	94.5	1130	0.9	69.1	1100	3.5	3.6	1126	1126.0	1.2	24.0	1090	1077.0	5.5	10.2
c109	1190	1190	1184.0	1180	0.5	55.4	1180	0.8	1190	0.0	227.6	1180	1170	0.8	143.2	1190	0.0	69.0	1180	0.8	2.5	1182	1182.0	0.7	27.5	1130	1117.0	6.1	11.7
c201	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.1	1810	0.0	44.9	1810	0.0	1.1	1810	1810.0	0.0	1.9	1810	1810.0	0.0	11.5
c202	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	1.9	1810	0.0	44.2	1810	0.0	1.1	1810	1810.0	0.0	0.9	1810	1810.0	0.0	12.5
c203	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.2	1810	1810	0.0	1.5	1810	0.0	42.0	1810	0.0	1.0	1810	1810.0	0.0	0.5	1810	1810.0	0.0	14.3
c204	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.5	1810	0.0	39.6	1810	0.0	1.0	1810	1810.0	0.0	0.2	1810	1810.0	0.0	17.1
c205	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.1	1810	0.0	42.0	1810	0.0	1.0	1810	1810.0	0.0	0.9	1810	1810.0	0.0	12.1
c206	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.1	1810	0.0	40.6	1810	0.0	1.0	1810	1810.0	0.0	0.2	1810	1810.0	0.0	13.1
c207	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.1	1810	0.0	40.8	1810	0.0	1.0	1810	1810.0	0.0	0.3	1810	1810.0	0.0	13.7
c208	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.1	1810	1810	0.0	0.1	1810	0.0	40.0	1810	0.0	0.8	1810	1810.0	0.0	0.2	1810	1810.0	0.0	14.3
r101	611	608	608.0	608	0.5	14.0	606	0.8	608	0.5	93.5	608	607	0.5	11.9	611	0.0	33.8	601	1.6	1.4	608	603.6	1.2	22.3	611	605.1	1.0	5.2
r102	843	837	837.0	837	0.7	47.9	843	0.0	837	0.7	162.6	822	814.6	2.5	23.0	843	0.0	45.2	807	4.3	1.7	834	828.0	1.8	19.9	814	805.7	4.4	7.6
r103	928	928	928.0	928	0.0	5.5	914	1.5	928	0.0	206.9	914	902	1.5	68.9	926	0.2	97.1	878	5.4	2.2	924	920.2	0.8	37.0	888	877.8	5.4	9.5
r104	975	972	969.6	969	0.6	51.5	951	2.5	969	0.6	209.1	939	939	3.7	103.0	964	1.1	84.7	941	3.5	3.8	967	961.0	1.4	36.3	925	914.7	6.2	11.0
r105	778	778	776.6	775	0.2	147.2	767	1.4	778	0.0	142.4	776	770.6	0.3	15.5	771	0.9	47.3	735	5.5	2.9	757	753.8	3.1	19.8	767	759.0	2.4	6.8
r106	906	906	903.8	895	0.2	24.7	868	4.2	906	0.0	185.9	884	878.2	2.4	54.7	905	0.1	78.7	870	4.0	3.5	905	897.6	0.9	30.9	885	861.3	4.9	8.4
r107	950	950	947.2	945	0.3	57.5	899	5.4	950	0.0	221.4	946	932.8	0.4	112.4	942	0.8	42.0	927	2.4	3.3	945	937.2	1.3	36.3	908	897.3	5.5	9.8
r108	995	991	991.0	991	0.4	79.3	964	3.1	994	0.1	220	968	958.6	2.7	198.5	977	1.8	70.6	982	1.3	3.2	970	967.0	2.8	46.7	947	929.6	6.6	11.1
r109	885	885	885.0	885	0.0	131.5	837	5.4	885	0.0	145.8	879	869.4	0.7	55.0	885	0.0	68.0	866	2.1	2.1	885	885.0	0.0	31.1	867	841.2	4.9	8.1
r110	915	915	907.6	900	0.8	50.8	898	1.9	915	0.0	168.6	915	893.2	0.0	92.9	893	2.4	39.5	870	4.9	2.0	890	889.0	2.8	24.3	843	835.6	8.7	8.3
r111	953	945	944.4	944	0.9	12.2	908	4.7	952	0.1	203.5	952	937.4	0.1	96.5	948	0.5	53.2	935	1.9	2.0	949	948.0	0.5	34.2	914	898.3	5.7	9.5
r112	974	971	967.4	964	0.7	69.1	956	1.8	967	0.7	251.8	968	950.4	0.6	184.6	958	1.6	40.5	939	3.6	3.1	957	948.0	2.7	28.8	916	898.8	7.7	8.8
r201	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	1.4	1458	0.0	42.0	1458	0.0	1.3	1458	1458.0	0.0	1.0	1458	1458.0	0.0	12.1
r202	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.1	1458	0.0	41.6	1458	0.0	1.1	1458	1458.0	0.0	0.5	1458	1458.0	0.0	12.6
r203	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.2	1458	0.0	40.2	1458	0.0	0.9	1458	1458.0	0.0	0.2	1458	1458.0	0.0	14.5
r204	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.1	1458	0.0	38.9	1458	0.0	0.6	1458	1458.0	0.0	0.1	1458	1458.0	0.0	16.2
r205	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.1	1458	0.0	39.6	1458	0.0	0.9	1458	1458.0	0.0	0.3	1458	1458.0	0.0	14.5
r206	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.1	1458	1458	0.0	0.2	1458	0.0	39.4	1458	0.0	0.9	1458	1458.0	0.0	0.2	1458	1458.0	0.0	15.2
r207	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.2	1458	0.0	38.7	1458	0.0	0.8	1458	1458.0	0.0	0.1	1458	1458.0	0.0	16.2
r208	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.1	1458	0.0	38.1	1458	0.0	0.5	1458	1458.0	0.0	0.1	1458	1458.0	0.0	17.2
r209	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.2	1458	0.0	39.6	1458	0.0	1.0	1458	1458.0	0.0	0.2	1458	1458.0	0.0	15.2
r210	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.2	1458	0.0	40.0	1458	0.0	0.9	1458	1458.0	0.0	0.2	1458	1458.0	0.0	14.8
r211	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.2	1458	1458	0.0	0.2	1458	0.0	38.7	1458	0.0	0.7	1458	1458.0	0.0	0.1	1458	1458.0	0.0	16.7
rc101	811	808	808.0	808	0.4	1.6	811	0.0	808	0.4	119.7	808	797.6	0.4	10.1	808	0.4	44.3	794	2.1	1.9	810	802.6	1.0	22.5	796	789.2	2.7	5.9
rc102	909	902	902.0	902	0.8	38.1	906	0.3	899	1.1	136.7	897	887	1.3	21.3	902	0.8	126.9	881	3.1	2.3	903	902.0	0.8	44.5	881	870.3	4.3	7.0
rc103	975	972	971.0	970	0.4	103.6	896	8.1	974	0.1	170.2	953	948.4	2.3	46.7	970	0.5	70.0	947	2.9	2.0	954	950.0	2.6	37.6	948	921.1	5.5	7.9
rc104	1065	1065	1064.6	1063	0.0	107.5	1017	4.5	1064	0.1	194.1	1048	1036	1.6	76.6	1059	0.6	75.3	1019	4.3	1.7	1059	1059.0	0.6	28.5	1021	997.0	6.4	8.4
rc105	875	875	875.0	875	0.0	64.5	867	0.9	875	0.0	127	875	866	0.0	20.3	875	0.0	46.2	841	3.9	1.5	862	861.8	1.5	20.2	861	850.0	2.9	6.9
rc106	909	909	908.8	908	0.0	37.6	826	9.1	909	0.0	143	902	896.2	0.8	20.7	901	0.9	41.4	874	3.9	2.5	900	897.4	1.3	34.7	876	854.9	6.0	6.7
rc107	987	980	975.8	972	1.1	75.9	980	0.7	980	0.7	155.4	966	963	2.1	46.0	980	0.7	93.8	951	3.6	1.9	980	977.2	1.0	26.0	945	928.1	6.0	7.2
rc108	1025	1023	1021.6	1020	0.3	60.8	1002	2.2	1020	0.5	173.2	1023	1015	0.2	53.6	1023	0.2	47.1	998	2.6	2.0	1025	1023.0	0.2	27.5	961	934.7	8.8	7.6
rc201	1724	1724	1724.0	1724	0.0	0.1	1724	0.0	1724	0.0	0.2	1724	1723	0.0	1.9	1724	0.0	41.7	1724	0.0	2.1	1724	1724.0	0.0	2.0	1724	1723.3	0.0	12.3
rc202	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.8	1724	0.0	41.0	1724	0.0	1.1	1724	1724.0	0.0	0.7	1724	1724.0	0.0	12.6
rc203	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.1	1724	0.0	39.8	1724	0.0	0.9	1724	1724.0	0.0	0.4	1724	1724.0	0.0	14.7
rc204	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.1	1724	0.0	38.7	1724	0.0	0.8	1724	1724.0	0.0	0.3	1724	1724.0	0.0	16.7

(*continued on next page*)

Table A4 (*continued*)

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t(s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
rc205	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	3.3	1724	0.0	41.6	1724	0.0	2.1	1724	1724.0	0.0	1.2	1724	1724.0	0.0	12.6
rc206	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.1	1724	0.0	39.8	1724	0.0	1.0	1724	1724.0	0.0	0.8	1724	1724.0	0.0	13.5
rc207	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.5	1724	0.0	39.7	1724	0.0	1.0	1724	1724.0	0.0	0.5	1724	1724.0	0.0	14.5
rc208	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.2	1724	1724	0.0	0.2	1724	0.0	38.9	1724	0.0	0.9	1724	1724.0	0.0	0.3	1724	1724.0	0.0	16.5
					0.3	29.3		1.2		0.1	103.4			0.6	45.1		0.3	49.7		1.7	1.7			0.6	15.6			2.7	11.5

Table A5
Results for Cordeau's problems for m = 1.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
pr01	308	308	308.0	308	0.0	0.0	308	0.0	305	1.0	20.8	308	307.2	0.0	1.2	305	1.0	8.3	304	1.3	0.5	308	307.0	0.3	5.8	308	306.7	0.4	1.0
pr02	404	404	404.0	404	0.0	103.1	404	0.0	394	2.5	47.9	404	403.6	0.0	3.7	404	0.0	29.1	385	4.7	0.6	392	391.2	3.2	4.9	392	386.6	4.3	2.1
pr03	394	394	392.8	388	0.3	0.6	388	1.5	394	0.0	72.9	388	388.0	1.5	4.1	394	0.0	59.9	384	2.5	1.0	394	394.0	0.0	12.6	392	387.9	1.5	2.7
pr04	489	489	488.2	487	0.2	81.6	475	2.9	489	0.0	109.3	489	475.4	0.0	14.7	489	0.0	106.7	447	8.6	1.9	489	486.0	0.6	67.4	489	466.1	4.7	4.4
pr05	595	592	592.0	592	0.5	68.5	591	0.7	594	0.2	185.4	587	578.0	1.3	20.5	589	1.0	281.7	576	3.2	4.6	589	586.8	1.4	91.4	579	570.1	4.2	7.5
pr06	590	583	576.6	563	2.3	16.3	586	0.7	590	0.0	189.9	590	584.2	0.0	29.0	575	2.5	253.4	538	8.8	2.5	576	573.6	2.8	247.4	553	538.4	8.7	7.0
pr07	298	298	298.0	298	0.0	0.1	288	3.4	298	0.0	26.5	298	297.0	0.0	1.7	298	0.0	15.0	291	2.3	0.4	298	298.0	0.0	4.3	298	296.6	0.5	1.2
pr08	463	463	463.0	463	0.0	0.1	463	0.0	454	1.9	77.4	463	463.0	0.0	3.8	462	0.2	76.0	463	0.0	1.0	463	462.0	0.2	15.0	463	460.7	0.5	3.1
pr09	493	493	493.0	493	0.0	30.5	493	0.0	490	0.6	137.8	493	482.0	0.0	12.3	482	2.2	102.3	461	6.5	1.4	482	481.8	2.3	126.3	483	468.9	4.9	4.6
pr10	594	594	569.2	562	4.2	106.3	568	4.4	568	4.4	222.2	579	564.4	2.5	32.7	578	2.7	189.7	539	9.3	3.6	574	570.4	4.0	210.3	568	549.3	7.5	8.0
pr11	353	353	353.0	353	0.0	3.7	351	0.6	353	0.0	30.8	330	329.0	6.5	1.9	351	0.6	10.3	330	6.5	0.3	353	350.6	0.7	1.7	338	329.4	6.7	1.1
pr12	442	441	441.0	441	0.2	0.6	440	0.5	433	2.0	59.8	435	435.0	1.6	6.5	430	2.7	26.3	431	2.5	0.9	432	432.0	2.3	23.5	436	429.9	2.7	2.5
pr13	467	457	457.0	457	2.1	13.2	458	1.9	466	0.2	89.5	453	452.4	3.0	16.2	452	3.2	49.0	450	3.6	1.9	461	458.1	1.9	8.9	451	440.2	5.7	3.4
pr14	567	552	552.0	552	2.6	6.6	543	4.2	521	8.1	144.4	549	540.6	3.2	32.4	540	4.8	134.3	482	15.0	1.1	565	560.1	1.2	76.6	526	505.0	10.9	5.1
pr15	708	707	688.0	682	2.8	54.2	649	8.3	707	0.1	248.2	665	656.6	6.1	29.4	666	5.9	118.5	638	9.9	5.3	670	666.4	5.9	274.6	667	655.7	7.4	9.8
pr16	674	650	620.4	613	8.0	21.7	588	12.8	619	8.2	228.6	673	643.4	0.1	60.9	616	8.6	558.0	559	17.1	4.1	614	613.4	9.0	205.1	594	580.0	13.9	9.1
pr17	362	362	361.6	360	0.1	53.8	362	0.0	360	0.6	34.7	359	345.6	0.8	5.4	362	0.0	37.6	346	4.4	0.2	359	359.0	0.8	6.6	360	351.3	3.0	1.5
pr18	539	539	537.6	532	0.3	55.5	535	0.7	497	7.8	99.0	535	530.8	0.7	10.4	539	0.0	61.8	479	11.1	0.8	535	535.0	0.7	22.1	511	485.2	10.0	3.7
pr19	562	490	482.6	464	14.1	120.5	543	3.4	538	4.3	164.6	510	507.8	9.3	22.1	531	5.5	152.9	499	11.2	2.7	544	541.0	3.7	70.7	513	499.9	11.0	5.9
pr20	667	648	645.8	637	3.2	7.1	629	5.7	588	11.8	202.7	662	655.0	0.7	57.0	626	6.1	475.3	570	14.5	2.5	608	604.8	9.3	346.9	628	610.2	8.5	9.9
					2.0	37.2		2.6		2.7	119.6			1.9	18.3		2.4	137.3		7.2	1.9			2.5	91.1			5.9	4.7

Table A6
Results for Cordeau's problems for m = 2.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg.	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
pr01	502	502	502.0	502	0.0	0.3	502	0.0	502	0.0	51.8	502	495.0	0.0	3.6	502	0.0	20.2	471	6.2	0.5	502	502.0	0.0	4.1	484	476.5	5.1	2.3
pr02	715	715	713.0	711	0.3	40.6	699	2.2	714	0.1	127.7	713	706.4	0.3	12.3	712	0.4	37.3	660	7.7	1.2	712	704.6	1.5	20.0	682	673.2	5.8	5.4
pr03	742	742	726.6	720	2.1	42.6	736	0.8	731	1.5	175.6	721	718.4	2.8	19.2	741	0.1	217.7	714	3.8	3.3	742	738.4	0.5	75.7	718	706.9	4.7	7.7
pr04	926	913	902.8	892	2.5	132.9	856	7.6	917	1.0	270.1	908	903.8	1.9	33.3	905	2.3	245.0	863	6.8	4.1	913	905.0	2.3	126.2	871	861.0	7.0	12.4
pr05	1101	1090	1086.4	1085	1.3	95.3	1036	5.9	1101	0.0	410.0	1082	1073.6	1.7	94.3	1053	4.4	249.8	1011	8.2	7.1	1052	1045.0	5.1	220.7	1023	1010.7	8.2	20.0
pr06	1076	1070	1050.4	1034	2.4	88.7	1014	5.8	1040	3.3	427.6	1076	1070.2	0.0	71.0	1022	5.0	439.3	997	7.3	9.8	1013	1009.8	6.2	445.7	980	960.2	10.8	21.0
pr07	566	566	566.0	566	0.0	4.1	561	0.9	566	0.0	71.8	566	557.8	0.0	6.0	566	0.0	20.5	552	2.5	1.0	561	559.2	1.2	6.7	560	554.9	2.0	3.5
pr08	834	829	829.0	829	0.6	8.6	810	2.9	824	1.2	184.1	829	824.4	0.6	20.0	822	1.4	75.6	796	4.6	5.1	810	806.6	3.3	74.6	810	795.5	4.6	8.6
pr09	909	907	904.0	902	0.6	98.1	887	2.4	878	3.4	304.9	901	883.4	0.9	49.7	854	6.1	120.7	867	4.6	5.2	865	859.2	5.5	225.3	837	822.2	9.5	13.7
pr10	1134	1128	1091.6	1060	3.7	43.9	1073	5.4	1117	1.5	447.0	1124	1109.0	0.9	81.5	1069	5.7	313.2	1004	11.5	10.3	1075	1067.4	5.9	297.9	1036	1011.9	10.8	22.8
pr11	566	566	564.8	564	0.2	16.0	559	1.2	559	1.2	71.3	547	539.4	3.4	3.7	566	0.0	17.5	542	4.2	0.7	561	558.2	1.4	2.1	547	525.4	7.2	2.5
pr12	774	765	764.4	762	1.2	12.5	750	3.1	768	0.8	143.6	767	750.8	0.9	28.0	759	1.9	39.0	727	6.1	1.3	757	754.8	2.5	19.4	749	717.5	7.3	6.1
pr13	843	**845**	840.4	833	0.3	73.5	803	4.7	832	1.3	238.6	829	827.0	1.7	41.3	825	2.1	107.0	757	10.2	2.4	831	831.0	1.4	39.5	764	746.0	11.5	9.2
pr14	1017	988	981.8	973	3.5	20.0	955	6.1	978	3.8	337.3	1009	999.0	0.8	117.3	922	9.3	111.7	925	9.0	8.1	946	939.8	7.6	120.9	929	882.5	13.2	13.6
pr15	1220	**1222**	1194.8	1158	2.1	131.9	1120	8.2	1205	1.2	479.1	1219	1210.4	0.1	126.6	1155	5.3	444.8	1126	7.7	8.2	1154	1145.4	6.1	356.1	1115	1077.9	11.6	23.1
pr16	1231	1183	1162.6	1150	5.6	130.0	1093	11.2	1124	8.7	500.5	1224	1217.8	0.6	209.6	1094	11.1	330.7	1110	9.8	11.0	1110	1106.4	10.1	708.8	1069	1047.6	14.9	24.2
pr17	652	646	645.2	644	1.0	95.7	628	3.7	639	2.0	117.0	630	626.0	3.4	9.6	643	1.4	20.2	624	4.3	1.3	652	652.0	0.0	16.3	617	601.1	7.8	3.8
pr18	953	945	933.2	927	2.1	58.3	910	4.5	937	1.7	231.0	927	914.4	2.7	39.5	929	2.5	88.2	877	8.0	2.9	933	929.0	2.5	77.3	860	835.3	12.4	9.5
pr19	1034	1019	1015.4	1010	1.8	146.2	967	6.5	1003	3.0	386.1	1018	1004.4	1.5	120.3	1017	1.6	302.7	955	7.6	5.5	1008	1004.2	2.9	277.5	893	871.6	15.7	16.3
pr20	1241	1207	1187.2	1180	4.3	76.8	1114	10.2	1155	6.9	541.6	1232	1224.4	0.7	128.5	1154	7.0	554.5	1056	14.9	10.7	1184	1173.3	5.5	600.5	1124	1061.5	14.5	26.2
					1.8	65.8		4.7		2.1	275.8			1.2	60.8		3.4	187.8		7.2	5.0			3.6	185.8			9.2	12.6

Table A7
Results for Cordeau's problems for m = 3.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
pr01	622	616	616.0	616	1.0	23.7	611	1.8	622	0.0	132.1	614	608.4	1.3	3.0	614	1.3	18.7	598	3.9	0.4	622	617.4	0.7	6.4	602	592.8	4.7	3.1
pr02	943	941	940.2	940	0.3	34.2	925	1.9	936	0.7	260.6	940	937.8	0.3	18.6	939	0.4	42.7	899	4.7	3.9	934	930.9	1.3	42.6	894	880.3	6.6	8.2
pr03	1010	1010	1008.6	1006	0.1	56.0	998	1.2	1010	0.0	301.5	1003	997.4	0.7	29.9	989	2.1	173.2	946	6.3	3.9	991	985.4	2.4	60.1	963	948.2	6.1	12.4
pr04	1294	1269	1266.6	1259	2.1	62.9	1197	7.5	1286	0.6	442.7	1294	1279.6	0.0	106.5	1253	3.2	168.7	1195	7.7	9.0	1258	1256.0	2.9	147.0	1200	1181.9	8.7	19.5
pr05	1482	1478	1468.8	1461	0.9	147.9	1423	4.0	1481	0.1	650.3	1480	1464	0.1	154.9	1431	3.4	226.6	1356	8.5	12.5	1438	1429.8	3.5	400.2	1364	1349.9	8.9	30.0
pr06	1514	1505	1481.4	1439	2.2	169.0	1373	9.3	1501	0.9	651.2	1512	1499.4	0.1	188.9	1469	3.0	332.4	1376	9.1	19.2	1462	1449.0	4.3	619.1	1362	1341.7	11.4	34.4
pr07	744	744	741.2	738	0.4	74.5	737	0.9	738	0.8	260.4	744	736.6	0.0	11.8	742	0.3	24.9	713	4.2	1.0	741	738.2	0.8	7.7	725	718.6	3.4	5.1
pr08	1139	1135	1129.2	1125	0.9	133.0	1106	2.9	1139	0.0	307.0	1132	1122.6	0.6	44.8	1131	0.7	157.1	1082	5.0	4.3	1132	1130.2	0.8	42.5	1074	1066.2	6.4	12.8
pr09	1282	1273	1269.2	1255	1.0	111.5	1171	8.7	1272	0.8	503.1	1268	1260.6	1.1	102.6	1236	3.6	251.3	1144	10.8	10.3	1215	1208.2	5.8	305.3	1175	1158.0	9.7	22.0
pr10	1573	**1574**	1556.4	1543	1.1	146.6	1434	8.8	1567	0.4	731.1	1573	1563.4	0.0	198.0	1477	6.1	574.5	1473	6.4	27.9	1465	1460.6	7.1	658.4	1433	1408.3	10.5	36.2
pr11	654	654	654.0	654	0.0	12.2	654	0.0	654	0.0	151.7	651	647	0.5	5.9	654	0.0	12.8	632	3.4	0.5	654	652.2	0.3	7.1	628	621.1	5.0	3.4
pr12	1002	990	987.4	985	1.5	95.5	971	3.1	997	0.5	294.3	979	975.2	2.3	61.0	967	3.5	43.0	902	10.0	1.8	973	966.0	3.6	26.3	947	929.8	7.2	8.7
pr13	1152	1147	1143.2	1142	0.8	115.5	1080	6.3	1145	0.6	378.9	1106	1102.2	4.0	77.0	1139	1.1	80.3	1046	9.2	8.2	1122	1111.0	3.6	62.7	1051	1026.4	10.9	13.7
pr14	1372	1339	1331.2	1324	3.0	152.0	1299	5.3	1315	4.2	533.7	1369	1357.4	0.2	130.9	1289	6.0	166.3	1197	12.8	8.3	1288	1279.4	6.7	140.9	1221	1200.7	12.5	20.9
pr15	1659	1654	1636.8	1592	1.3	106.3	1508	9.1	1654	0.3	708.1	1609	1594.2	3.0	246.6	1550	6.6	414.2	1488	10.3	14.6	1592	1588.4	4.3	443.0	1501	1474.5	11.1	33.7
pr16	1668	1629	1606.2	1582	3.7	183.0	1521	8.8	1609	3.5	818.1	1668	1654.6	0.0	413.5	1530	8.3	697.6	1478	11.4	28.2	1534	1521.2	8.8	700.1	1471	1442.5	13.5	38.3
pr17	841	837	828.2	824	1.5	112.2	837	0.5	841	0.0	184.3	832	822.4	1.1	15.2	838	0.4	38.0	808	3.9	0.9	832	826.6	1.7	19.6	785	773.1	8.1	5.5
pr18	1282	1256	1243.8	1225	3.0	161.3	1248	2.7	1276	0.5	386.6	1281	1270.8	0.1	63.5	1262	1.6	164.1	1165	9.1	6.0	1253	1241.2	3.2	88.6	1152	1099.7	14.2	14.1
pr19	1417	1391	1372.8	1342	3.1	137.2	1270	10.4	1403	1.0	604.1	1417	1393.6	0.0	216.3	1329	6.2	220.7	1238	12.6	10.2	1353	1340.2	5.4	310.5	1199	1181.4	16.6	25.8
pr20	1690	1671	1658.6	1645	1.9	162.2	1510	10.7	1658	1.9	909.7	1684	1677.6	0.4	277.4	1593	5.7	681.3	1514	10.4	18.2	1601	1587.8	6.0	677.6	1481	1451.4	14.1	39.2
					1.5	109.8		5.2		0.8	460.5			0.8	118.3		3.2	224.4		8.0	9.5			3.7	238.3			9.5	19.4

Table A8
Results for Cordeau's problems for m = 4.

		ES					CP		I3CH			GVNS				SSA			ILS			ABC				GRILS			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)	Best	Avg	Gap	t (s)
pr01	657	657	657.0	657	0.0	0.5	657	0.0	657	0.0	0.1	657	655.8	0.0	3.6	657	0.0	15.5	644	2.0	0.2	657	655.8	0.2	3.5	654	652.7	0.7	3.9
pr02	1079	1075	1067.8	1062	1.0	85.5	1063	1.5	1073	0.6	380.6	1070	1064	0.8	25.6	1069	0.9	44.0	1014	6.0	2.4	1058	1053.6	2.4	38.0	1019	1003.4	7.0	10.4
pr03	1246	**1247**	1236.0	1221	0.8	145.0	1198	3.9	1232	1.1	436.6	1215	1210	2.5	57.2	1201	3.6	156.1	1162	6.7	10.5	1199	1198.4	3.8	74.7	1135	1124.4	9.8	16.4
pr04	1585	1566	1549.8	1541	2.2	176.8	1486	6.2	1585	0.0	603.6	1531	1526	3.4	137.2	1535	3.2	393.2	1452	8.4	11.6	1529	1525.0	3.8	199.9	1444	1431.7	9.7	26.9
pr05	1844	1840	1818.6	1801	1.4	138.6	1665	9.7	1838	0.3	902.9	1827	1811	0.9	208.4	1759	4.6	321.0	1665	9.7	19.6	1761	1755.4	4.8	330.7	1682	1647.7	10.6	41.1
pr06	1886	1861	1851.4	1840	1.8	222.2	1680	10.9	1835	2.7	939.6	1855	1840	1.6	309.9	1839	2.5	721.0	1696	10.1	35.4	1775	1763.6	6.5	678.3	1707	1666.7	11.6	49.4
pr07	876	872	866.4	855	1.1	142.3	871	0.6	872	0.5	228.9	870	862.6	0.7	17.9	871	0.6	31.6	840	4.1	1.6	859	855.2	2.4	18.5	835	822.6	6.1	6.4
pr08	1385	**1392**	1372.6	1364	0.9	102.6	1309	5.5	1377	0.6	429.9	1361	1358	1.7	67.3	1358	1.9	118.8	1267	8.5	6.9	1349	1336.8	3.5	87.1	1285	1278.1	7.7	17.3
pr09	1619	1603	1585.8	1571	2.1	151.9	1495	7.7	1604	0.9	698.5	1607	1595	0.7	171.1	1565	3.3	252.5	1460	9.8	13.8	1543	1532.8	5.3	176.4	1479	1454.6	10.2	31.2
pr10	1943	1936	1887.8	1857	2.8	211.8	1784	8.2	1943	0.0	1044.4	1916	1904	1.4	275.1	1853	4.6	502.0	1782	8.3	38.7	1848	1842.0	5.2	725.7	1794	1740.8	10.4	50.4
pr11	657	657	657.0	657	0.0	0.2	657	0.0	657	0.0	0.1	657	657	0.0	5.0	657	0.0	13.5	654	0.5	0.2	657	657.0	0.0	0.4	657	654.3	0.4	4.0
pr12	1132	1117	1108.6	1100	2.1	105.4	1090	3.7	1120	1.1	477.1	1123	1110	0.8	36.7	1116	1.4	134.2	1041	8.0	1.9	1132	1130.1	0.2	38.7	1066	1047.6	7.5	10.9
pr13	1386	1377	1350.4	1336	2.6	148.5	1279	7.7	1386	0.0	672.0	1334	1325	3.8	76.6	1355	2.2	112.6	1263	8.9	6.6	1342	1332.4	3.9	70.2	1250	1237.5	10.7	17.7
pr14	1674	1664	1638.6	1617	2.1	245.0	1553	7.2	1651	1.4	783.2	1648	1637	1.6	243.6	1586	5.3	192.5	1528	8.7	16.6	1581	1569.2	6.3	211.5	1517	1489.9	11.0	28.5
pr15	2065	2007	1999.0	1992	3.2	171.7	1881	8.9	2065	0.0	1161.7	1946	1933	5.8	428.3	1929	6.6	518.6	1818	12.0	19.5	1940	1937.6	6.2	450.5	1869	1823.3	11.7	45.7
pr16	2065	1999	1967.6	1926	4.7	197.1	1828	11.5	2017	2.3	1183.8	2014	2000	2.5	632.0	1921	7.0	769.7	1889	8.5	35.9	1928	1924.0	6.8	685.0	1822	1778.2	13.9	53.2
pr17	934	931	927.2	922	0.7	227.1	927	0.7	934	0.0	332.8	921	914.6	1.4	14.0	926	0.9	30.5	889	4.8	1.9	932	926.6	0.8	21.3	879	867.7	7.1	6.8
pr18	1539	1496	1489.4	1481	3.2	79.8	1496	2.8	1539	0.0	559.5	1518	1505	1.4	93.0	1455	5.5	104.8	1352	12.2	5.7	1474	1462.8	5.0	86.5	1357	1329.1	13.6	17.9
pr19	1760	1750	1723.8	1700	2.1	166.7	1543	12.3	1750	0.6	919.4	1723	1688	2.1	334.1	1695	3.7	277.6	1560	11.4	22.2	1680	1670.8	5.1	285.4	1511	1478.6	16.0	34.0
pr20	2062	**2082**	2027.8	1994	1.7	261.5	1902	7.8	2062	0.0	1196.6	2019	2012	2.1	463.1	1901	7.8	685.8	1846	10.5	26.9	1966	1958.8	5.0	826.3	1852	1810.3	12.2	53.7
					1.8	149.0		5.8		0.6	647.6			1.8	180.0		3.3	269.8		8.0	13.9			3.8	250.4			9.4	26.3

Table A9
Results for Vansteenwegen problems based on Solomon's instances.

		ES					CP		I3CH			GVNS			SSA			ILS			ABC			
Prob.	Opt	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)
c101	1810	1810	1810.0	1810	0.0	0.1	1810	0.0	1810	0.0	1.3	1754.0	3.1	27	1770	2.2	89.5	1720	5.0	4.1	1794	1786.0	1.3	59.1
c102	1810	1810	1810.0	1810	0.0	3.5	1810	0.0	1810	0.0	3.6	1794.0	0.9	23.2	1810	0.0	52.9	1790	1.1	4.2	1810	1810.0	0.0	14.3
c103	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.7	1810.0	0.0	5.7	1810	0.0	52.9	1810	0.0	3	1810	1810.0	0.0	6.4
c104	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	1.6	1810	0.0	44.4	1810	0.0	1.8	1810	1810.0	0.0	1.9
c105	1810	1810	1810.0	1810	0.0	0.3	1810	0.0	1810	0.0	2.8	1810.0	0.0	1.6	1780	1.7	152.7	1770	2.2	2.8	1791	1782.0	1.5	58.8
c106	1810	1810	1810.0	1810	0.0	0.4	1810	0.0	1810	0.0	178.9	1806.0	0.2	9.1	1800	0.6	123.1	1750	3.3	3.8	1789	1786.0	1.3	53.2
c107	1810	1810	1810.0	1810	0.0	0.1	1810	0.0	1810	0.0	9.1	1810.0	0.0	1.1	1760	2.8	61.1	1790	1.1	3.1	1789	1784.0	1.4	53.2
c108	1810	1810	1810.0	1810	0.0	0.2	1810	0.0	1810	0.0	230.6	1810.0	0.0	0.7	1770	2.2	59.6	1810	0.0	2.5	1797	1792.0	1.0	44.2
c109	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.7	1810.0	0.0	0.1	1810	0.0	63	1810	0.0	2	1810	1810.0	0.0	24.1
c201	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.5	1810.0	0.0	0.1	1810	0.0	45	1810	0.0	1.1	1810	1810.0	0.0	1.9
c202	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	1.8	1810	0.0	44.3	1810	0.0	1.1	1810	1810.0	0.0	0.9
c203	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	1.5	1810	0.0	42.1	1810	0.0	1	1810	1810.0	0.0	0.5
c204	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.7	1810.0	0.0	0.5	1810	0.0	39.8	1810	0.0	1	1810	1810.0	0.0	0.2
c205	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.5	1810.0	0.0	0.1	1810	0.0	42	1810	0.0	1	1810	1810.0	0.0	0.9
c206	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	0.1	1810	0.0	40.7	1810	0.0	1	1810	1810.0	0.0	0.2
c207	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	0.1	1810	0.0	40.9	1810	0.0	1	1810	1810.0	0.0	0.3
c208	1810	1810	1810.0	1810	0.0	0.0	1810	0.0	1810	0.0	0.6	1810.0	0.0	0.1	1810	0.0	40.1	1810	0.0	0.8	1810	1810.0	0.0	0.2
r101	1458	1458	1458.0	1458	0.0	7.7	1458	0.0	1458	0.0	1.2	1432.2	1.8	23.4	1455	0.2	67.9	1441	1.2	2.5	1458	1457.4	0.0	25.1
r102	1458	1458	1458.0	1458	0.0	61.9	1458	0.0	1458	0.0	1.9	1441.2	1.2	20.7	1458	0.0	84.9	1450	0.5	3.1	1458	1455.4	0.2	18.4
r103	1458	1458	1456.2	1455	0.1	40.9	1458	0.0	1458	0.0	958.1	1446.6	0.8	23.5	1455	0.2	59.6	1450	0.5	2	1458	1455.0	0.2	14.3
r104	1458	1452	1449.4	1445	0.6	138.0	1439	1.3	1454	0.3	1014.3	1418.2	2.7	55	1442	1.1	131.3	1402	3.8	2.3	1440	1437.8	1.4	59
r105	1458	1458	1457.0	1453	0.1	87.5	1458	0.0	1458	0.0	44.2	1441.6	1.1	32.3	1458	0.0	136.5	1435	1.6	4.1	1458	1458.0	0.0	43.2
r106	1458	1458	1458.0	1458	0.0	82.8	1458	0.0	1458	0.0	84	1437.6	1.4	25.3	1458	0.0	116.3	1441	1.2	3.1	1458	1458.0	0.0	32.4
r107	1458	1453	1452.8	1452	0.4	87.6	1444	1.0	1458	0.0	1069.5	1435.0	1.6	46	1452	0.4	78.8	1431	1.9	3.3	1458	1452.0	0.4	45.8
r108	1458	1458	1457.2	1456	0.1	99.4	1446	0.8	1456	0.1	1543.5	1441.8	1.1	56.4	1447	0.8	87.1	1430	1.9	2.7	1455	1451.8	0.4	64.9
r109	1458	1458	1455.6	1449	0.2	131.3	1455	0.2	1458	0.0	474.2	1433.4	1.7	42.8	1453	0.3	128.1	1432	1.8	2.5	1458	1449.4	0.6	64.3
r110	1458	1454	1451.4	1448	0.5	66.8	1443	1.0	1454	0.3	1281.1	1433.4	1.7	50.1	1454	0.3	172.9	1419	2.7	4.4	1451	1449.0	0.6	54.8
r111	1458	1456	1452.8	1450	0.4	189.9	1458	0.0	1458	0.0	571.8	1430.2	1.9	40	1444	1.0	114.6	1410	3.3	3	1452	1449.8	0.6	47.5
r112	1458	1456	1452.6	1447	0.4	85.1	1440	1.2	1456	0.1	3488.4	1434.4	1.6	58.7	1446	0.8	78	1418	2.7	2.4	1448	1448.0	0.7	43.1
r201	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.7	1458.0	0.0	1.4	1458	0.0	42.2	1458	0.0	1.3	1458	1458.0	0.0	1
r202	1458	1458	1457.4	1455	0.0	21.9	1458	0.0	1458	0.0	5.5	1450.2	0.5	19.8	1458	0.0	58.5	1443	1.0	2.7	1458	1458.0	0.0	5.8
r203	1458	1458	1458.0	1458	0.0	0.3	1458	0.0	1458	0.0	0.7	1458.0	0.0	2.8	1458	0.0	39.4	1458	0.0	1.6	1458	1458.0	0.0	0.8
r204	1458	1458	1458.0	1458	0.0	43.7	1452	0.4	1458	0.0	35.7	1452.6	0.4	6.6	1447	0.8	70.2	1440	1.2	3.4	1458	1453.4	0.3	35.6
r205	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.8	1458.0	0.0	0.4	1458	0.0	38.7	1458	0.0	1.1	1458	1458.0	0.0	1.3
r206	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.7	1458.0	0.0	0.7	1458	0.0	38.4	1458	0.0	1.1	1458	1458.0	0.0	0.6
r207	1458	1458	1455.4	1453	0.2	90.6	1443	1.0	1456	0.1	948.8	1449.8	0.6	12.4	1452	0.4	155.9	1428	2.1	1.7	1454	1448.4	0.7	35.7
r208	1458	1458	1458.0	1458	0.0	0.4	1458	0.0	1458	0.0	4.5	1458.0	0.0	0.9	1457	0.1	54.2	1458	0.0	1.6	1458	1457.8	0.0	12.6
r209	1458	1458	1458.0	1458	0.0	0.0	1458	0.0	1458	0.0	0.7	1458.0	0.0	0.2	1458	0.0	38.8	1458	0.0	1.1	1458	1458.0	0.0	0.9
r210	1458	1458	1458.0	1458	0.0	0.1	1458	0.0	1458	0.0	0.8	1458.0	0.0	0.9	1458	0.0	39	1458	0.0	1.2	1458	1458.0	0.0	0.6
r211	1458	1458	1457.0	1455	0.1	95.5	1451	0.5	1449	0.6	908.3	1452.6	0.4	14.6	1451	0.5	66.2	1422	2.5	1.9	1456	1451.2	0.5	29.3
rc101	1724	1722	1719.4	1713	0.3	88.7	1724	0.0	1724	0.0	54.2	1690.2	2.0	28.1	1712	0.7	79.9	1686	2.2	4.3	1717	1713.6	0.6	52.9
rc102	1724	1716	1712.8	1708	0.6	80.1	1718	0.3	1724	0.0	125.3	1685.0	2.3	30.1	1718	0.3	103.4	1659	3.8	2.9	1713	1705.2	1.1	38.8
rc103	1724	1724	1723.4	1721	0.0	171.9	1724	0.0	1724	0.0	7.1	1709.0	0.9	37.9	1724	0.0	82.6	1689	2.0	3.4	1724	1721.0	0.2	40.9
rc104	1724	1724	1724.0	1724	0.0	2.1	1724	0.0	1724	0.0	8.7	1718.0	0.3	43.6	1719	0.3	87.7	1719	0.3	3.2	1724	1723.4	0.0	43.7
rc105	1724	1719	1715.2	1709	0.5	150.5	1719	0.3	1724	0.0	31.2	1689.8	2.0	32.9	1716	0.5	87	1691	1.9	3.9	1724	1716.0	0.5	47.6
rc106	1724	1719	1718.0	1714	0.3	119.8	1716	0.5	1724	0.0	54.2	1690.6	1.9	40.5	1714	0.6	63.7	1665	3.4	4.8	1723	1706.0	1.0	50.5
rc107	1724	1724	1724.0	1724	0.0	14.2	1724	0.0	1724	0.0	12.8	1718.4	0.3	51.5	1722	0.1	51.4	1701	1.3	2.4	1724	1724.0	0.0	46
rc108	1724	1724	1724.0	1724	0.0	45.2	1719	0.3	1724	0.0	165.6	1713.0	0.6	51.1	1719	0.3	121	1698	1.5	5.6	1724	1720.6	0.2	47.5
rc201	1724	1724	1724.0	1724	0.0	0.1	1724	0.0	1724	0.0	0.8	1724.0	0.0	1.9	1724	0.0	41.7	1724	0.0	2.1	1724	1724.0	0.0	2
rc202	1724	1724	1720.4	1714	0.2	91.7	1724	0.0	1719	0.3	1516	1702.8	1.2	10.3	1714	0.6	47.8	1686	2.2	1.7	1719	1717.4	0.4	12.7
rc203	1724	1724	1724.0	1724	0.0	0.7	1724	0.0	1724	0.0	0.8	1724.0	0.0	2.2	1724	0.0	38.7	1724	0.0	2.9	1724	1724.0	0.0	2.1
rc204	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.8	1724.0	0.0	0.1	1724	0.0	37.9	1724	0.0	1	1724	1724.0	0.0	1.1

(*continued on next page*)

Table A9 (*continued*)

		ES					CP		I3CH			GVNS			SSA			ILS			ABC			
Prob.	Opt	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)
rc205	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.7	1723.0	0.1	3.2	1724	0.0	41.4	1724	0.0	2.1	Missing			
rc206	1724	1724	1724.0	1724	0.0	0.6	1724	0.0	1724	0.0	0.8	1724.0	0.0	2.5	1724	0.0	45.4	1708	0.9	1.3	1724	1724.0	0.0	4.3
rc207	1724	1724	1724.0	1724	0.0	0.5	1724	0.0	1724	0.0	0.8	1724.0	0.0	2.2	1724	0.0	39.7	1713	0.6	1.5	1724	1724.0	0.0	1.9
rc208	1724	1724	1724.0	1724	0.0	0.0	1724	0.0	1724	0.0	0.9	1724.0	0.0	0.2	1724	0.0	38.4	1724	0.0	1.1	1724	1724.0	0.0	1.1
					0.1	37.5		0.2		0.0	265.2		0.6	16.9		0.3	70.3		1.1	2.4			0.3	24.6

Table A10
Results for Vansteenwegen problems based on Cordeau's instances.

		ES					CP		I3CH			GVNS			SSA			ILS			**ABC**			
Prob.	BKS	Best	Avg.	Worst	Gap	t (s)	Best	Gap	Best	Gap	t (s)	Avg	Gap	t (s)	Best	Gap	t (s)	Best	Gap	t (s)	Best	Avg	Gap	t (s)
pr01	657	619	615.8	614	6.3	102.0	622	5.3	619	5.8	146.7	608.4	7.4	3.1	614	6.5	18.5	608	7.5	0.7	623	617.4	6.0	6.4
pr02	1220	1209	1206.0	1199	1.1	155.2	1198	1.8	1207	1.1	669.7	1198.8	1.7	27.9	1205	1.2	48.0	1180	3.3	4.5	1215	1203.8	1.3	32.3
pr03	1788	1774	1770.0	1766	1.0	94.5	1775	0.7	1781	0.4	1383.7	1760.8	1.5	50.2	1758	1.7	124.2	1738	2.8	11.3	1769	1764.6	1.3	138.6
pr04	2477	2477	2473.8	2470	0.1	83.1	2468	0.4	2477	0.0	641.9	2467.4	0.4	114.0	2461	0.6	393.3	2428	2.0	45.4	2477	2474.2	0.1	266.7
pr05	3351	3351	3351.0	3351	0.0	45.1	3351	0.0	3351	0.0	19.3	3351.0	0.0	43.2	3351	0.0	1109.4	3297	1.6	37.3	3351	3351.0	0.0	331.1
pr06	3671	3671	3670.8	3670	0.0	143.3	3671	0.0	3671	0.0	30.0	3670.6	0.0	196.4	3661	0.3	1646.7	3650	0.6	106.1	3671	3670.6	0.0	752.0
pr07	948	940	938.2	937	1.0	100.4	942	0.6	943	0.5	299.7	935.0	1.4	13.8	948	0.0	27.2	909	4.1	1.5	936	931.4	1.8	14.9
pr08	2006	2006	2006.0	2006	0.0	41.9	2006	0.0	2006	0.0	55.9	2004.6	0.1	46.9	2006	0.0	128.5	1984	1.1	12.0	2006	2006.0	0.0	37.7
pr09	2736	2736	2736.0	2736	0.0	11.6	2736	0.0	2736	0.0	10.8	2736.0	0.0	8.5	2736	0.0	794.8	2729	0.3	33.0	2736	2736.0	0.0	104.8
pr10	3850	3850	3850.0	3850	0.0	6.3	3850	0.0	3850	0.0	9.1	3850.0	0.0	9.4	3847	0.1	1369.2	3850	0.0	52.3	3850	3850.0	0.0	126.8
					1.0	78.3		0.9		0.8	326.7		1.2	51.3		1.0	566.0		2.3	30.4			1.1	181.1

Appendix B. New best-known solutions

Solomon's problems for m = 2

r209

Profit: 1417

Tour1: 27 69 63 64 11 62 88 31 52 47 19 82 83 5 98 44 14 38 86 16 61 85 99 94 87 57 22 40 53 26 3 78 34 50 20 70 1 35 66 32 48 17 60 89 13 58

Tour2: 95 59 42 15 2 23 67 39 75 72 21 28 12 76 29 79 33 9 71 65 81 51 30 10 49 36 8 45 84 93 96 97 37 100 43 54 4 74 56 25 55 24 80 68 77

Cordeau's problems for m = 2

pr13

Profit: 845

Tour1: 109 107 87 14 80 26 139 110 27 31 44 124 2 94 48 130 23 81 90 7 32 86

Tour2: 138 66 53 82 29 33 128 25 22 123 18 16 104 43 63 5 28 35 39 17 91 129 10 112 79 70 74 141

pr15

Profit: 1222

Tour1: 150 11 235 29 165 153 54 18 238 67 71 164 176 40 216 47 147 2 215 27 26 179 237 43 204 123 85 182 136 62 77 198 64 212 79 32 170 118 5

Tour2: 146 145 3 107 205 112 185 74 24 66 181 230 28 135 197 80 187 203 142 81 217 102 129 157 193 84 21 48 168

Cordeau's problems for m = 3

pr10

Profit: 1574

Tour1: 261 143 166 108 263 206 64 50 229 280 138 189 238 281 65 77 9 260 112 160 233 207 144 173 177 17 129 51 161 69

Tour2: 178 170 79 185 109 61 25 219 284 114 97 267 181 232 205 249 35 86 126 100 179 118 62 14 54 184 279 116

Tour3: 39 196 11 220 191 142 87 275 20 34 42 2 115 158 243 27 72 155 242 71 101 59 285 156 208 6 110 128 152

Cordeau's problems for m = 4

pr03

Profit: 1247

Tour1: 118 39 17 87 139 88 52 80 129 91 26 71 35 107 109 78 14 10 112 12

Tour2: 115 101 105 37 51 58 68 6 117 56 24 70 79 135 125 61 111 34 19 4 25

Tour3: 130 122 15 28 5 104 86 65 29 53 82 59 93 33 128 123 18 22 43 63 144

Tour4: 7 64 38 99 23 81 32 138 96 90 134 2 124 44 94 48 142 77

pr08

Profit: 1392

Tour1: 74 72 43 20 98 17 75 44 54 135 56 24 65 59 123 13 47 141 101

Tour2: 63 129 84 93 114 115 127 97 106 133 34 109 10 128 113 8 16 23 28 140 19

Tour3: 136 30 100 83 36 52 58 48 131 55 92 107 21 85 1 33 41 144 71 111

Tour4: 117 87 37 108 68 77 125 32 76 64 112 14 6 70 67 124 60 138

pr20

Profit: 2082

Tour1: 92 18 192 144 260 9 112 160 118 179 100 108 122 244 98 62 246 80 64 50 206 14 54 65 184 77 103 138 123 263 233 251 177 190 143 207

Tour2: 161 146 51 35 86 126 249 180 119 267 97 25 114 219 284 124 129 277 17 22 166 32 168 228 227 273 271

Tour3: 7 155 44 243 27 72 158 185 115 109 221 181 232 205 49 20 134 34 42 141 275 87 94 69 43 2 258 6 16 71 110

Tour4: 46 21 1 208 222 242 56 101 150 105 89 174 285 196 245 11 39 191 91 178 142 36 154 52 96 254

References

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies – A comprehensive introduction. *Natural Computing, 1*(1), 3–52.

Bouly, H., Dang, D.-C., & Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. *4OR, 8*(1), 49–70.

Chao, I.-M. (1993). *Algorithms and solutions to multi-level vehicle routing problems.* Doctoral DissertationMD, USA: University of Maryland at College Park College Park.

Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996a). The team orienteering problem. *European Journal of Operational Research, 88*(3), 464–474.

Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996b). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research, 88*(3), 475–489.

Cordeau, J.-F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks, 30*, 105–119.

Cura, T. (2014). An artificial bee colony algorithm approach for the team orienteering. *Computers & Industrial Engineering, 74*, 270–290.

Eiben, A., & Smith, J. E. (2003). *Introduction to evolutionary computing.* Berlin: Springer-Verlag Berlin Heidelberg.

Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2019). Efficient cluster-based heuristics for the team orienteering problem with time windows. *Asia-Pacific Journal of Operational Research, 36*(1), 1–44.

Gedik, R., Kirac, E., Milburn, A. B., & Rainwater, C. (2017). A constraint programming approach for the team orienteering problem with time windows. *Computers &*

Industrial Engineering, 107, 178–195.

Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics, 34*, 307–318.

Golden, B. L., Wang, Q., & Liu, L. (1988). A multifaceted heuristic for the orienteering problem. *Naval Research Logistics, 35*(3), 359–366.

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research, 255*, 315–322.

Gunawan, A., Lau, H. C., Vansteenwegen, P., & Lu, K. (2017). Well-tuned algorithms for the Team Orienteering Problem with Time Windows. *Journal of the Operational Research Society, 68*(8), 861–876.

Hu, W., Fathi, M., & Pardalos, P. M. (2018). A multi-objective evolutionary algorithm based on decomposition and constraint programming for the multi-objective team orienteering problem with time windows. *Applied Soft Computing, 73*, 383–393.

Hu, Q., & Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research, 232*, 276–286.

IOF. (n.d.). History: International Orienteering Federation. Retrieved November 24, 2017, from International Orienteering Federation: http://orienteering.org/about-the-iof/history/.

Kantor, M. G., & Rosenwein, M. B. (1992). The orienteering problem with time windows. *The Journal of the Operational Research Society, 43*(6), 629–635.

Karabulut, K., & Tasgetiren, M. F. (2013). A discrete artificial bee colony algorithm for the team orienteering problem with time windows. *Proceedings of the IEEE workshop on computational intelligence in production and logistics systems (CIPLS), Singapore* (pp. 99–106). .

Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering, 54*(3), 648–665.

Ke, L., Guo, H., & Zhang, Q. (2014). *A cooperative approach between metaheuristic and branch-and-price for the team orienteering problem with time windows. IEEE congress on evolutionary computation (CEC). Beijing, China*.

Keller, P. C. (1989). Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research, 41*(2), 224–231.

Kurz, M. E., & Askin, R. G. (2001). Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research, 39*(16), 3747–3769.

Labadie, N., Mansini, R., Melechovský, J., & Calvo, R. W. (2012). The team orienteering problem with time windows: An LP-based granular variable neighboorhood search. *European Journal of Operational Research, 220*, 15–27.

Labadie, N., Melechovský, J., & Calvo, R. W. (2011). Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics, 17*(6), 729–753.

Lin, S.-W., & Yu, V. F. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research, 217*, 94–107.

Lü, Z., Glover, F., & Hao, J.-K. (2010). A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research, 207*(3), 1254–1262.

Mansini, R., Pelizzari, M., & Calvo, W. R. (2006). A Granular Variable Neighborhood Search for the Tour Orieentering Problem with Time Windows. Technical Report of the Department of Electronics for Automation, University of Brescia, RT.

Montemanni, R., & Gambardella, L. M. (2009). An ant colony system for team orienteering problem with time windows. *Foundations of Computing and Decision Sciences, 34*(4), 287–306.

Montemanni, R., Weyland, D., & Gambardella, L. M. (2011). *An enhanced ant colony system for the team orienteering problem with time windows. International Symposium on Computer Science and Society*.

Montgomery, D. C. (2012). *Design and analysis of experiments* (8th ed.). Wiley.

Ramesh, R., & Brown, K. M. (1991). An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research, 18*(2), 151–165.

Rechenberg, I. (1971). *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis)*. Technical University of Berlin, Department.

Righini, G., & Salani, M. (2006). *Dynamic programming for the orienteering problem with time windows.* Milano: Dipartimento di Tecnologie dell'Informazione, Universita degli Studi Milano.

Righini, G., & Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming. *Computers & Operations Research, 36*(4), 1191–1203.

Santini, A. (2019). An adaptive large neighbourhood search algorithm for the orienteering problem. *Expert Systems with Applications, 123*(1), 154–167.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics, 159*(2), 139–171.

Schwefel, H.-P. (1975). Evolutionsstrategie und numerische Optimierung (PhD thesis). Technical University of Berlin.

Snyder, L. V., & Daskin, M.s. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research, 174*(1), 38–53.

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

Souffriau, W., Vansteenwegen, P., Berghe, G. V., & Van Oudheusden, D. (2010). A Path Relinking approach for the Team Orienteering Problem. *Computers & Operations Research, 37*(11), 1853–1859.

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2011). The multiconstraint team orienteering problem with multiple time windows. *Transportation Science, 47*(1), 53–63.

Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G. V., & Oudheusden, D. V. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence, 22*(10), 964–985.

Tae, H., & Kim, B.-I. (2015). A branch-and-price approach for the team orienteering problem with time windows. *International Journal of Industrial Engineering, 22*(2), 243–251.

Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research, 32*(6), 1379–1407.

Tasgetiren, M. F. (2002). A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic & Social Research, 4*(2), 1–26.

Tasgetiren, M. F., & Smith, A. E. (2000). *A genetic algorithm for the orienteering problem. Proceedings of the 2000 congress on evolutionary computation. La Jolla, CA*.

Tasgetiren, M. F., Suganthan, P. N., & Pan, Q.-K. (2007). A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. *GECCO '07 Proceedings of the 9th annual conference on Genetic and evolutionary computation* (pp. 158–167). London, England: ACM.

Tasgetiren, M. F., Suganthan, P. N., & Pan, Q.-K. (2010). An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation, 215*(9), 3356–3368.

Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research, 37*, 351–367.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *The Journal of the Operational Research Society, 35*(9), 797–809.

Vansteenwegen, P., & Oudheusden, D. V. (2007). The mobile tourist guide: An OR opportunity. *OR Insight, 20*(3), 21–27.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2009a). Iterated local search for the team orienteering problem with time windows. *Computers and Operations Research, 36*(12), 3281–3290.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2009b). Metaheuristics for Tourist trip planning. *Metaheuristics in the Service Industry. Lecture Notes in Economics and Mathematical Systems* (pp. 15–31). Heidelberg: Springer, Berlin.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2009c). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research, 196*(1), 118–127.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research, 209*, 1–10.

Wang, X., Golden, B. L., & Wasil, E. A. (2008). Using a genetic algorithm to solve the generalized orienteering problem. *The vehicle routing problem: Latest advances and new challenges* (pp. 263–274). US: Springer.

Yu, V. F., Jewpanya, P., Lin, S.-W., & Redi, A. P. (2019). Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering, 127*, 213–224.