4OR-Q J Oper Res (2010) 8:49–70
DOI 10.1007/s10288-008-0094-4

RESEARCH PAPER

A memetic algorithm for the team orienteering problem

Hermann Bouly · Duc-Cuong Dang · Aziz Moukrim

Received: 8 February 2008 / Revised: 24 November 2008 / Published online: 7 January 2009

Abstract The team orienteering problem (TOP) is a generalization of the orienteering problem. A limited number of vehicles is available to visit customers from a potential set. Each vehicle has a predefined running-time limit, and each customer has a fixed associated profit. The aim of the TOP is to maximize the total collected profit. In this paper we propose a simple hybrid genetic algorithm using new algorithms dedicated to the specific scope of the TOP: an Optimal Split procedure for chromosome evaluation and local search techniques for mutation. We have called this hybrid method a memetic algorithm for the TOP. Computational experiments conducted on standard benchmark instances clearly show our method to be highly competitive with existing ones, yielding new improved solutions in at least 5 instances.

Keywords Selective vehicle routing problem · Memetic algorithm · Optimal split · Metaheuristic · Destruction/construction

MSC classification (2000) 90B06 · 90C27 · 90C59

This paper is an extension of a preliminary work published in EvoWorkshops 2008 proceedings (Bouly et al. 2008).

H. Bouly (✉) · D.-C. Dang · A. Moukrim
Université de Technologie de Compiègne Heudiasyc,
CNRS UMR 6599, BP 20529, 60205 Compiègne, France
e-mail: hermann.bouly@hds.utc.fr

D.-C. Dang
e-mail: duc-cuong.dang@hds.utc.fr

A. Moukrim
e-mail: aziz.moukrim@hds.utc.fr

H. Bouly
VEOLIA Environnement, Direction de la Recherche,
17/19, rue La Pérouse, 75016 Paris, France

0 Introduction

The team orienteering problem (TOP) first appeared in Butt and Cavalier (1994) under the name of the Multiple Tour Maximum Collection Problem. The term TOP, first introduced in Chao et al. (1996), comes from a sporting activity: team orienteering. A team consists of several members who all begin at the same starting point. Each member tries to collect as many reward points as possible within a certain time before reaching the finishing point. Available points can be awarded only once. Chao et al. (1996) also created a set of instances, used nowadays as standard benchmark instances for the TOP.

The TOP is an extension to multiple-vehicle of the orienteering problem (OP), also known as the selective traveling salesman problem (STSP). The TOP is also a generalization of vehicle routing problems (VRPs) where only a subset of customers can be serviced. As an extension of these problems, the TOP clearly appears to be NP-hard.

The assumption shared by problems of the TSP and VRPs family is that all customers should be serviced. In many real applications this assumption is not valid. In practical conditions, it is not always possible to satisfy all customer orders within a single time period. Shipping of these orders needs to be spread over different periods and, in some cases, as a result of uncertainty or dynamic components, customers may remain unserviced, meaning that the problem has a selective component which companies need to address. Different real applications have been shown to be corresponding to the TOP in the literature, such as the problem of college football player recruiting (Chao et al. 1996) and the technician routing and scheduling problem (Tang and Miller-Hooks 2005).

Recently Feillet et al. (2005) have reviewed the TOP as an extension of TSPs with profits. They focus both on travel costs and selection of customers, given a fixed fleet size. They discuss and show that minimizing travel costs and maximizing profits are opposite criteria. Most of the metaheuristics shown to be effective for the TOP are variable neighborhood search (VNS) (Archetti et al. 2006) and, more recently, ants colony optimization (ACO) (Ke et al. 2008). The memetic algorithm (MA), first introduced by Moscato (1999), is a recent technique that has been shown to be competitive for VRPs (Prins 2004). An MA consists in a combination of an evolutionary algorithm with local search (LS) methods. In this paper we propose an MA that makes use of an Optimal Split procedure developed for the specific case of the TOP. An Optimal Split is performed using a modified version of the program evaluation and review technique/critical path method (PERT/CPM). The Critical Path has been solved using dynamic programming, first introduced by Bellman (1957). We have also developed a strong heuristic for population initialization that we have termed Iterative Destruction/Construction Heuristic (IDCH). It is based on Destruction/Construction principles described in Ruiz and Stützle (2007), combined with a priority rule and LS. Computational results are compared with those of different methods corresponding to the early work of Chao et al. (1996), Tang and Miller-Hooks (2005), the best method of the literature, proposed by Archetti et al. (2006), and the most recent methods, proposed by Khemakhem et al. (2007) and Ke et al. (2008).

The article is organized as follows. Section 1 gives a formal description of the TOP. Section 2 describes our algorithm with employed heuristics, an adaptation of the PERT/CPM method yielding an Optimal Split procedure and the MA design.

Numerical results on standard instances are presented in Sect. 3. At the end we put forward some conclusions.

1 Problem formulation

The TOP can be modeled with a graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the vertex set representing customers, and $E = \{(i, j) \mid i, j \in V\}$ is the edge set. Each vertex i is associated with a profit P_i. There is also a *departure* and an *arrival* vertex, denoted, respectively, d and a. A tour r is represented as an ordered list of $|r|$ customers from V: $r = (r_1, \ldots, r_{|r|})$. Each *tour* begins at the departure vertex and ends at the arrival vertex. We denote the total profit collected from a tour r as $P(r) = \sum_{i \in r} P_i$, and the total travel cost or duration $C(r) = C_{d,r_1} + \sum_{i=1}^{i=|r|-1} C_{r_i, r_{i+1}} + C_{r_{|r|},a}$, where $C_{i,j}$ denotes the travel cost between i and j. Travel costs are assumed to satisfy the triangle inequality. The fleet is composed of m identical vehicles. So, a *solution* is a set of m (or fewer) feasible tours in which each customer is visited only once. A tour r is feasible if its length does not exceed a pre-defined limited running length L. That means a solution is feasible if $C(r) \leq L$ for any tour r. The goal is to find a collection of m (or fewer) tours that maximizes the total profit while satisfying the pre-specified tour length limit L on each tour.

2 Resolution methods

Genetic algorithms (GA) are classified as Evolutionary Algorithms: a *population* of solutions evolves through the repetitive combination of its *individuals*. A GA *encodes* each solution into a similar structure called a *chromosome*. An encoding is said to be *indirect* if a decoding procedure is necessary to extract solutions from chromosomes. In this paper we use a simple indirect encoding that we denote as a *giant tour*, and an Optimal Split procedure as the decoding process. Optimal Split was first introduced by Beasley (1983) and Ulusoy (1985), respectively, for the node routing and arc routing problems. The splitting procedure we propose here is specific to the TOP.

To insert a chromosome in the population and to identify improvements, it is necessary to know the performance of each individual in the population through an *evaluation* procedure. In our algorithm, this evaluation involves the splitting procedure corresponding to chromosome decoding. The combining of two chromosomes to produce a new one is called *crossover*. A diversification process is also used to avoid homogeneity in the population. This diversification is obtained through a *mutation* operation and through conditions on the insertion of new chromosomes in the population.

The MA is a combination of an evolutionary algorithm and LS techniques. This combination has been shown to be effective for the VRP in Prins (2004). Our MA is a combination of GA and some LS techniques.

2.1 Chromosome encoding and evaluation

As mentioned above, we do not directly encode a solution, but an ordered list of all the customers in V, which we term a *giant tour*. To evaluate the individual performance

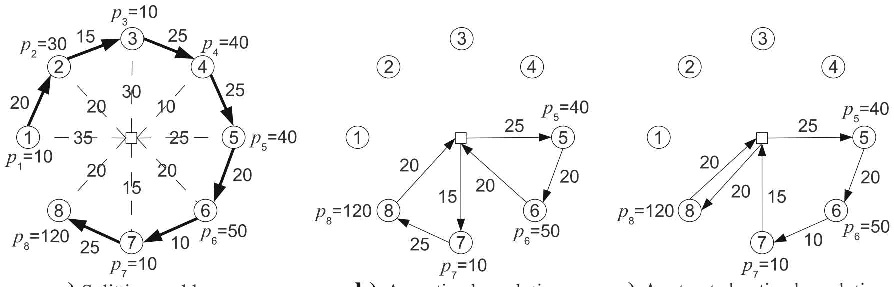


Fig. 1 A giant tour and two optimal π-solutions for $n = 8$, $m = 2$ and $L = 70$

of a chromosome it is necessary to split the giant tour to identify the multiple-vehicle solution and unrouted customers.

The giant tour is encoded as a *sequence*, i.e. a permutation of V that we denote as π. We extract m tours from the giant tour while respecting the order of the customers in the sequence and the constraint on the length of each tour (referred to from now on as the L-constraint). We consider only tours whose customers are adjacent in the sequence, so that a tour can be identified by its starting point i in the sequence and the number of customers following i in π, denoted $l_i \geq 0$, to be included in the tour. A tour corresponds to the subsequence $(\pi[i], \ldots, \pi[i + l_i])$ and is denoted as $\langle i, l_i \rangle_\pi$.

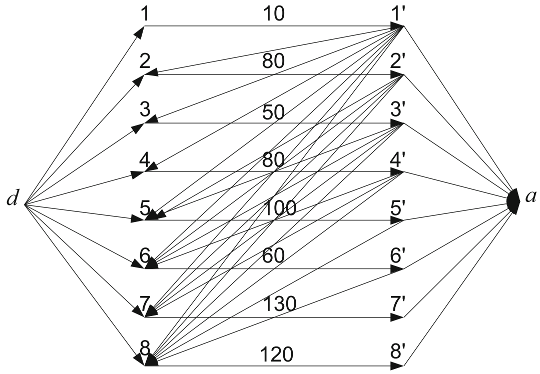
The maximum possible value of l_i for a feasible tour, given a sequence π, depends on L. A tour of maximum length is called a *saturated* tour, meaning that all customers following i in π are included in the tour as long as the L-constraint is satisfied, or until the end of the sequence is reached. Customers remaining unrouted after splitting can only be located between tours in π. We denote as $l_i^{\max,\pi}$ the number of customers following i in the sequence starting with $\pi[i]$ such that $\langle i, l_i^{\max,\pi} \rangle_\pi$ is saturated, i.e. the tour represented by $\langle i, l_i^{\max,\pi} + 1 \rangle_\pi$ is infeasible, or the end of the sequence has been reached.

A π-solution $S_\pi = (\langle i_1, l_{i_1} \rangle_\pi, \ldots, \langle i_k, l_{i_k} \rangle_\pi)$ is such that $k \leq m$, $\langle i_p, l_{i_p} \rangle_\pi$ respects the L-constraint for each p, and $i_{q+1} > i_q + l_q$ for each q in $1, \ldots, k - 1$. A π-solution is optimal if the sum of the profits from customers in the subsequences, denoted $P(S_\pi)$, is such that there exists no π-solution yielding a greater profit. A π-solution $(\langle i_1, l_{i_1} \rangle_\pi, \ldots, \langle i_k, l_{i_k} \rangle_\pi)$ is said to be saturated if each tour $\langle i_p, l_{i_p} \rangle_\pi$ in S_π is saturated for $p < k$. Figure 1 describes an instance with eight customers. Profits from these customers are, respectively, 10, 30, 10, 40, 40, 50, 10, 120. We consider $\pi = (1, 2, 3, 4, 5, 6, 7, 8)$. Two optimal π-solutions, one of which is saturated, are shown in Fig. 1.

The splitting problem consists in identifying a π-solution that maximizes the collected profit. We made the proof that an optimal splitting of the giant tour is obtained through consideration of only the saturated tours. This is formalized in Proposition 1.

Proposition 1 *For any instance of the TOP where m is the maximum number of vehicles available, for any sequence π of customers of this instance and for any optimal π-solution S_π including k tours ($k \leq m$), it does exist a saturated optimal π-solution S'_π including k' tours, with $k' \leq k$, so that $P(S'_\pi) = P(S_\pi)$.*

Fig. 2 Graph representation for the splitting problem



Proof Demonstrating this proposition can be done by iteratively replacing a non-saturated tour by another saturated tour without changing the total profit until a saturated π-solution has been produced. □

Therefore, the splitting can be done considering only saturated tours. Consequently, we are only interested in finding saturated π-solutions.

2.1.1 Optimal evaluation

The splitting problem can be formulated as finding a path on an acyclic graph $H = (X, F)$ with the set of vertices $X = \{d, 1, 1', 2, 2', \ldots, n, n', a\}$. An arc linking nodes x and x' represents a saturated tour starting with customer $\pi[x]$. The weight $w_{x,x'}$ of this arc is set to the value of the collected profit of the corresponding tour. An arc linking nodes x' and y with $y > x + l_x$ shows that the tour starting with $\pi[y]$ can commence after the tour starting with $\pi[x]$. These arcs are weighted by $w_{x',y} = 0$. This construction allows any set of q compatible saturated tours can be interpreted as a path on the graph. Furthermore, that path passes through exactly $2q + 1$ arcs.

Figure 2 shows the graph corresponding to the splitting problem in Fig. 1. Values of $l_i^{\max,\pi}$ for each starting customer i are 0, 2, 1, 1, 2, 1, 1, 0.

The splitting problem is finding the longest path in the new graph H that does not use more than $2m + 1$ arcs (m is the maximum number of vehicles). This can be done by modifying the well-known PERT/CPM method as follows.

For each node k in H (except the departure node d) we create two arrays μ_k and γ_k of fixed size $2m + 1$. Component $\mu_k[i]$ memorizes the maximum profit collected within a path of i arcs long from d to k and $\gamma_k[i]$ memorizes the predecessor of k matching the corresponding maximum profit. As H does not have any cycle, the idea is to visit nodes from d to a and to fill the two arrays μ_k and γ_k for each node k. At the end of the procedure, the largest component of μ_a represents the maximum profit that can be reached by splitting the sequence. Next, a backtrack is performed on γ_k in order to determine the corresponding tours.

Nodes are visited in the order $1, 1', 2, 2', \ldots, n, n', a$. We denote as $\Gamma^-(i)$ the set of the predecessors of i in H. We compute $\mu_k[i]$, the component i of the vector μ at node k, as follows: $\mu_k[i] = \max_{j \in \Gamma^-(k)}\{\mu_j[i-1] + w_{j,i}\}$. At the end of the procedure,

Table 1 Detailed results of different methods on benchmark instances set number 1 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
*p*1.2.*b*	15	15	15	15	15	15	15	15	15	15	15	15
*p*1.2.*c*	20	20	20	20	20	20	20	20	20	20	20	20
*p*1.2.*d*	30	30	30	30	30	30	30	30	30	30	30	30
*p*1.2.*e*	45	45	45	45	45	45	45	45	45	45	45	45
*p*1.2.*f*	80	80	80	80	80	80	80	80	80	80	80	80
*p*1.2.*g*	90	90	90	90	90	90	90	90	90	90	90	90
*p*1.2.*h*	110	110	110	110	110	110	110	110	110	110	110	110
*p*1.2.*i*	135	135	135	135	130	135	135	135	135	135	135	135
*p*1.2.*j*	155	155	155	155	155	155	155	155	155	155	155	155
*p*1.2.*k*	175	175	175	175	175	175	175	175	175	175	175	175
*p*1.2.*l*	195	195	195	195	190	195	195	195	195	195	195	195
*p*1.2.*m*	215	215	215	215	215	215	215	215	215	215	215	215
*p*1.2.*n*	235	235	235	235	235	235	235	235	235	235	235	235
*p*1.2.*o*	240	240	240	240	240	240	240	240	240	240	240	240
*p*1.2.*p*	250	250	250	250	250	250	250	250	250	250	250	–
*p*1.2.*q*	265	265	265	265	265	265	265	265	265	265	265	–
*p*1.2.*r*	280	280	280	280	280	280	280	280	280	280	280	–
*p*1.3.*c*	15	15	15	15	15	15	15	15	15	15	15	15
*p*1.3.*d*	15	15	15	15	15	15	15	15	15	15	15	15
*p*1.3.*e*	30	30	30	30	30	30	30	30	30	30	30	30
*p*1.3.*f*	40	40	40	40	40	40	40	40	40	40	40	40
*p*1.3.*g*	50	50	50	50	50	50	50	50	50	50	50	50
*p*1.3.*i*	105	105	105	105	70	70	105	105	105	105	105	105
*p*1.3.*j*	115	115	115	115	100	105	115	115	115	115	115	115
*p*1.3.*k*	135	135	135	135	135	135	135	135	135	135	135	135
*p*1.3.*l*	155	155	155	155	150	150	155	155	155	155	155	155
*p*1.3.*m*	175	175	175	175	175	175	175	175	175	175	175	175
*p*1.3.*n*	190	190	190	190	190	190	190	190	190	190	190	190
*p*1.3.*p*	220	220	220	220	220	220	220	220	220	220	220	220
*p*1.3.*q*	230	230	230	230	230	230	230	230	230	230	230	230
*p*1.4.*d*	15	15	15	15	15	15	15	15	15	15	15	15
*p*1.4.*e*	15	15	15	15	15	15	15	15	15	15	15	15
*p*1.4.*f*	25	25	25	25	25	25	25	25	25	25	25	25
*p*1.4.*g*	35	35	35	35	35	35	35	35	35	35	35	35
*p*1.4.*h*	45	45	45	45	45	45	45	45	45	45	45	45
*p*1.4.*i*	60	60	60	60	60	60	60	60	60	60	60	60
*p*1.4.*j*	75	75	75	75	75	75	75	75	75	75	75	75
*p*1.4.*k*	100	100	100	100	100	100	100	100	100	100	100	100
*p*1.4.*l*	120	120	120	120	120	120	120	120	120	120	120	120
*p*1.4.*m*	130	130	130	130	130	130	130	130	130	130	130	130

Table 1 continued

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p1.4.n$	155	155	155	155	155	155	155	155	155	155	155	155
$p1.4.o$	165	165	165	165	165	165	165	165	165	165	165	165
$p1.4.p$	175	175	175	175	175	175	175	175	175	175	175	175
$p1.4.q$	190	190	190	190	190	190	190	190	190	190	190	190
$p1.4.r$	210	210	210	210	210	210	210	210	210	210	210	210

the greatest value of μ_a indicates the longest path length, corresponding to the highest profit that can be reached in the original problem. We can use array γ to rebuild that path, and finally translate non-zero weighted links into their corresponding tours. The complexity of this modified PERT/CPM method for Optimal Splitting of the giant tour is $O(m \cdot n^2)$.

2.1.2 Fast evaluation

We may also use a faster evaluation technique, which we call Quick Split, as a splitting procedure. This simple split uses the assumption that if the tour p ends with the customer $\pi[x]$, the next tour begins with customer $\pi[x+1]$. This assumption that there are no unrouted customers between two different tours in the sequence and considering only saturated tours enables us to obtain an approximate evaluation where the first tour begins with $i_1 = 1$. The complexity of this method is $O(n)$.

2.2 Local search as mutation operator

To complete the MA, LS techniques are used as mutation operators with probability pm. We use different neighborhoods during mutation. The procedure starts with an unmarked neighborhood, which is chosen at random. To accelerate searching procedure, an improvement is applied as soon as it has been found. Each time a neighborhood has been unsuccessfully scanned, it is marked. When an improvement is found, the procedure restarts by unmarking all neighborhoods before choosing randomly a new one. Mutation operator is stopped when all neighborhoods are marked.

Many neighborhoods are tested on a large set of experiments until we found the following combination of three ones, giving most performance for our algorithm:

Shift operator. Each customer is extracted from the sequence π and its insertion in all other positions is evaluated.

Swap operator. An exchange of all couples of customers i and j in the sequence is evaluated.

Table 2 Detailed results of different methods on benchmark instances set number 2 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p2.2.a$	90	90	90	90	90	90	90	90	90	90	90	90
$p2.2.b$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.2.c$	140	140	140	140	140	140	140	140	140	140	140	140
$p2.2.d$	160	160	160	160	160	160	160	160	160	160	160	160
$p2.2.e$	190	190	190	190	190	190	190	190	190	190	190	190
$p2.2.f$	200	200	200	200	200	200	200	200	200	200	200	200
$p2.2.g$	200	200	200	200	200	200	200	200	200	200	200	200
$p2.2.h$	230	230	230	230	230	230	230	230	230	230	230	230
$p2.2.i$	230	230	230	230	230	230	230	230	230	230	230	230
$p2.2.j$	260	260	260	260	260	260	260	260	260	260	260	260
$p2.2.k$	275	275	275	275	275	275	275	275	275	275	275	275
$p2.3.a$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.3.b$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.3.c$	105	105	105	105	105	105	105	105	105	105	105	105
$p2.3.d$	105	105	105	105	105	105	105	105	105	105	105	105
$p2.3.e$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.3.f$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.3.g$	145	145	145	145	145	145	145	145	145	145	145	145
$p2.3.i$	200	200	200	200	200	200	200	200	200	200	200	200
$p2.3.j$	200	200	200	200	200	200	200	200	200	200	200	200
$p2.3.k$	200	200	200	200	200	200	200	200	200	200	200	200
$p2.4.a$	10	10	10	10	10	10	10	10	10	10	10	10
$p2.4.b$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.4.c$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.4.d$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.4.e$	70	70	70	70	70	70	70	70	70	70	70	70
$p2.4.f$	105	105	105	105	105	105	105	105	105	105	105	105
$p2.4.g$	105	105	105	105	105	105	105	105	105	105	105	105
$p2.4.h$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.4.i$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.4.j$	120	120	120	120	120	120	120	120	120	120	120	120
$p2.4.k$	180	180	180	180	180	180	180	180	180	180	180	180

In TSP problems, each neighbor obtained using the two operators described above is evaluated before a movement is carried out, leading to $O(n^2)$. That is not possible for the TOP since the aim is to evaluate the new profit and not saved travel costs. As the evaluation using PERT/CPM has a complexity of $O(m \cdot n^2)$, and to keep a complexity of $O(n^3)$ for all LS techniques used in mutation, we decided to use Quick Split in association with these two operators. A *compressed* version of the current chromosome is

Table 3 Detailed results of different methods on benchmark instances set number 3 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p3.2.a$	90	90	90	90	90	90	90	90	90	90	90	90
$p3.2.b$	150	150	150	150	150	150	150	150	150	150	150	150
$p3.2.c$	180	180	180	180	180	180	180	180	180	180	180	180
$p3.2.d$	220	220	220	220	220	220	220	220	220	220	220	220
$p3.2.e$	260	260	260	260	260	260	260	260	260	260	260	260
$p3.2.f$	300	300	300	300	300	300	300	300	300	300	300	300
$p3.2.g$	360	360	360	360	360	360	360	360	360	360	360	360
$p3.2.h$	410	410	410	410	400	410	410	410	410	410	410	410
$p3.2.i$	460	460	460	460	450	460	460	460	460	460	460	460
$p3.2.j$	510	510	510	510	510	510	510	510	510	510	510	510
$p3.2.k$	550	550	550	550	550	550	550	550	550	550	550	550
$p3.2.l$	590	590	590	590	570	590	590	590	590	590	590	–
$p3.2.m$	620	620	620	620	620	620	620	620	620	620	620	–
$p3.2.n$	660	660	660	660	650	660	660	660	660	660	660	–
$p3.2.o$	690	690	690	690	690	690	690	690	690	690	690	–
$p3.2.p$	720	720	720	720	720	720	720	720	720	720	720	–
$p3.2.q$	760	760	760	760	760	760	760	760	760	760	760	–
$p3.2.r$	790	790	790	790	790	790	790	790	790	790	790	–
$p3.2.s$	800	800	800	800	800	800	800	800	800	800	800	–
$p3.2.t$	800	800	800	800	800	800	800	800	800	800	800	800
$p3.3.a$	30	30	30	30	30	30	30	30	30	30	30	30
$p3.3.b$	90	90	90	90	90	90	90	90	90	90	90	90
$p3.3.c$	120	120	120	120	120	120	120	120	120	120	120	120
$p3.3.d$	170	170	170	170	170	170	170	170	170	170	170	170
$p3.3.e$	200	200	200	200	200	200	200	200	200	200	200	200
$p3.3.f$	230	230	230	230	230	230	230	230	230	230	230	230
$p3.3.g$	270	270	270	270	270	270	270	270	270	270	270	270
$p3.3.h$	300	300	300	300	300	300	300	300	300	300	300	300
$p3.3.i$	330	330	330	330	330	330	330	330	330	330	330	330
$p3.3.j$	380	380	380	380	380	380	380	380	380	380	380	380
$p3.3.k$	440	440	440	440	440	440	440	440	440	440	440	440
$p3.3.l$	480	480	480	480	480	480	480	480	480	480	480	480
$p3.3.m$	520	520	520	520	520	520	520	520	520	520	520	520
$p3.3.n$	570	570	570	570	570	570	570	570	570	570	570	570
$p3.3.o$	590	590	590	590	590	590	590	590	590	590	590	590
$p3.3.p$	640	640	640	640	640	640	640	640	640	640	640	640
$p3.3.q$	680	680	680	680	680	680	680	680	680	680	680	680
$p3.3.r$	710	710	710	710	710	710	710	710	710	710	710	710
$p3.3.s$	720	720	720	720	720	720	720	720	720	720	720	–
$p3.3.t$	760	760	760	760	760	760	760	760	760	760	760	–

Table 3 continued

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
*p*3.4.*a*	20	20	20	20	20	20	20	20	20	20	20	20
*p*3.4.*b*	30	30	30	30	30	30	30	30	30	30	30	30
*p*3.4.*c*	90	90	90	90	90	90	90	90	90	90	90	90
*p*3.4.*d*	100	100	100	100	100	100	100	100	100	100	100	100
*p*3.4.*e*	140	140	140	140	140	140	140	140	140	140	140	140
*p*3.4.*f*	190	190	190	190	190	190	190	190	190	190	190	190
*p*3.4.*g*	220	220	220	220	220	220	220	220	220	220	220	220
*p*3.4.*h*	240	240	240	240	240	240	240	240	240	240	240	240
*p*3.4.*i*	270	270	270	270	270	270	270	270	270	270	270	270
*p*3.4.*j*	310	310	310	310	310	310	310	310	310	310	310	310
*p*3.4.*l*	380	380	380	380	380	380	380	380	380	380	380	380
*p*3.4.*m*	390	390	390	390	390	390	390	390	390	390	390	390
*p*3.4.*n*	440	440	440	440	440	440	440	440	440	440	440	440
*p*3.4.*o*	500	500	500	500	500	500	500	500	500	500	500	500
*p*3.4.*p*	560	560	560	560	560	560	560	560	560	560	560	560
*p*3.4.*q*	560	560	560	560	560	560	560	560	560	560	560	560
*p*3.4.*r*	600	600	600	600	600	600	600	600	600	600	600	600
*p*3.4.*s*	670	670	670	670	670	670	670	670	670	670	670	670
*p*3.4.*t*	670	670	670	670	670	670	670	670	670	670	670	670

produced before entering LS by left-shifting identified tours to the beginning of the sequence. It allows solutions produced by Shift operator and the Swap operator to be evaluated by the Quick Split (see Sect. 2.1.2) quickly and efficiently. In order to restore a *standard* version of chromosome at the end of these LS techniques, when an improving neighbor has been selected, unrouted customers are redistributed along the chromosome in the same order as in the initial chromosome.

Destruct and Repair operator. The idea is to remove a small part of the solution with a view to rebuilding an improved solution (see Ruiz and Stützle 2007). This LS is applied to the solution given by PERT/CPM method on the current chromosome. A certain number (selected randomly between 1 and n/m) of customers are removed from tours and redeclared as unrouted customers. The solution is reconstructed using a parallel version of the Best Insertion algorithm (Solomon 1987). This constructive method evaluates the insertion cost $(C_{i,z} + C_{z,j} - C_{i,j})/P_z$ of any unrouted customer z between any couple of customers i and j in a tour r so that j directly follows i in r. The feasible insertion that minimizes the cost is then processed, and the method loops back to the evaluation of the remaining unrouted customers. If more than one possible insertion minimizes the insertion cost, one of them is chosen at random. This process is iterated until no further insertions are feasible, either because no tour can accept

Table 4 Detailed results of different methods on benchmark instances set number 4 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p4.2.a$	206	206	206	206	206	206	206	206	206	206	206	206
$p4.2.b$	338.7	341	341	341	327	341	341	341	341	341	341	341
$p4.2.c$	447.9	452	452	452	447	452	452	452	452	452	452	452
$p4.2.d$	527.5	531	528	531	521	522	531	531	531	531	531	531
$p4.2.e$	596.9	618	618	618	594	611	618	618	618	618	618	618
$p4.2.f$	672.6	687	678	687	660	672	678	681	687	687	687	–
$p4.2.g$	736.8	757	757	757	735	746	756	756.7	757	757	757	–
$p4.2.h$	818.2	827	835	835	779	811	810	822	835	835	835	–
$p4.2.i$	894.1	918	918	918	850	891	918	918	918	918	918	–
$p4.2.j$	953.2	965	962	962	918	945	962	963.3	964	965	965	–
$p4.2.k$	1001.1	1022	1022	1022	984	1002	1013	1016	1022	1022	1022	–
$p4.2.l$	1063.5	1071	1071	1074	1022	1052	1069	1070.3	1071	1074	1074	–
$p4.2.m$	1110.6	1130	1126	1132	1098	1102	1125	1129.7	1132	1132	1132	–
$p4.2.n$	1146.9	1168	1167	1174	1142	1151	1170	1172.7	1174	1174	1174	–
$p4.2.o$	1175.8	1215	1218	1218	1184	1184	1217	1217	1217	1218	1218	–
$p4.2.p$	1215	1242	1241	1241	1222	1233	1237	1240	1242	1242	1242	–
$p4.2.q$	1234.3	1263	1263	1263	1254	1261	1263	1265	1267	1265	1267	–
$p4.2.r$	1263.4	1288	1285	1285	1283	1287	1291	1291.3	1292	1288	1292	–
$p4.2.s$	1288.4	1304	1300	1301	1303	1304	1304	1304	1304	1304	1304	–
$p4.2.t$	1304.4	1306	1306	1306	1306	1306	1306	1306	1306	1306	1306	–
$p4.3.b$	38	38	38	38	38	38	38	38	38	38	38	38
$p4.3.c$	193	193	193	193	193	193	193	193	193	193	193	193
$p4.3.d$	333	335	335	335	332	335	335	335	335	335	335	335
$p4.3.e$	463.2	468	468	468	455	460	468	468	468	468	468	468
$p4.3.f$	569.2	579	579	579	567	571	579	579	579	579	579	579
$p4.3.g$	651.6	653	653	653	632	646	653	653	653	653	653	653
$p4.3.h$	712.6	720	727	729	692	712	717	722.3	725	729	729	729
$p4.3.i$	779.2	796	807	809	770	785	807	808.3	809	809	809	809
$p4.3.j$	839.4	861	857	861	828	841	856	859	861	861	861	–
$p4.3.k$	895.7	918	918	919	896	911	919	919	919	919	919	–
$p4.3.l$	954.2	979	975	979	931	970	972	972.7	974	979	979	–
$p4.3.m$	1023.1	1053	1053	1062	1008	1039	1039	1049.3	1063	1063	1063	–
$p4.3.n$	1100.3	1121	1114	1121	1071	1116	1114	1118.7	1121	1121	1121	–
$p4.3.o$	1158.1	1170	1170	1172	1128	1150	1167	1169.7	1172	1172	1172	–
$p4.3.p$	1201.7	1221	1197	1222	1190	1218	1205	1216	1222	1222	1222	–
$p4.3.q$	1227.4	1252	1243	1245	1227	1235	1245	1248.7	1252	1252	1252	–
$p4.3.r$	1255.7	1267	1267	1273	1261	1268	1269	1270.7	1273	1273	1273	–
$p4.3.s$	1283.7	1293	1277	1295	1294	1295	1295	1295	1295	1295	1295	–
$p4.3.t$	1302.3	1305	1293	1304	1300	1304	1304	1304	1304	1305	1305	–

Table 4 continued

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p4.4.d$	38	38	38	38	38	38	38	38	38	38	38	38
$p4.4.e$	183	183	183	183	183	183	183	183	183	183	183	183
$p4.4.f$	324	324	324	324	324	324	324	324	324	324	324	324
$p4.4.g$	460.1	461	461	461	461	461	461	461	461	461	461	461
$p4.4.h$	552	571	571	571	556	563	571	571	571	571	571	571
$p4.4.i$	641.6	657	657	657	646	657	657	657	657	657	657	657
$p4.4.j$	726.7	732	732	732	707	721	732	732	732	732	732	732
$p4.4.k$	814.2	821	821	821	793	808	821	821	821	821	821	821
$p4.4.l$	868.4	880	879	880	846	873	879	879	879	880	880	–
$p4.4.m$	904.7	918	915	918	901	912	916	916.7	918	919	919	–
$p4.4.n$	946.3	961	968	976	933	947	962	964.7	969	977	977	–
$p4.4.o$	1001.1	1036	1051	1061	1023	1053	1051	1057.7	1061	1061	1061	–
$p4.4.p$	1074	1111	1119	1120	1096	1105	1110	1114.7	1124	1120	1124	–
$p4.4.q$	1106.2	1145	1157	1161	1135	1149	1161	1161	1161	1161	1161	–
$p4.4.r$	1168.7	1200	1206	1207	1191	1204	1207	1210	1216	1211	1216	–
$p4.4.s$	1233.9	1249	1248	1260	1246	1253	1255	1257.7	1259	1260	1260	–
$p4.4.t$	1268.4	1281	1278	1285	1268	1278	1281	1282.3	1284	1285	1285	–

additional customers, or because all customers are routed (the solution is optimal in this case). The complexity is $O(n^3)$, since all customer insertions in all positions have to be evaluated and the process is iterated at most n times to insert all customers.

2.3 Algorithm initialization

To create some good solutions for an initial population, we developed an Iterative Destruction/Construction Heuristic (IDCH) based on the *Destruct and Repair operator* and some diversification components. The key idea of this heuristic is that the more difficult it is to insert an unrouted customer into a solution, the more this customer will be considered for insertion.

Starting with an empty solution, we use the parallel version of the Best Insertion (Solomon 1987) to build a first solution. On following iterations a small part of the current solution is destroyed by removing a limited random number of customers (1, 2 or 3) from tours, and a 2-opt procedure is used to reduce the travel cost of tours. A reconstruction phase is then processed using a parallel prioritized version of the Best Insertion. The destruction and construction phases are iterated, and each time a customer remains unrouted after the construction phase its priority is increased by the value of its associated profit. At each construction phase the subset of unrouted customers with the highest priority is considered for insertion. When no more of these customers can be inserted, unrouted customers with lower priorities are considered,

Table 5 Detailed results of different methods on benchmark instances set number 5 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
p5.2.b	20	20	20	20	20	20	20	20	20	20	20	20
p5.2.c	50	50	50	50	50	50	50	50	50	50	50	50
p5.2.d	80	80	80	80	80	80	80	80	80	80	80	80
p5.2.e	180	180	180	180	180	180	180	180	180	180	180	180
p5.2.f	240	240	240	240	240	240	240	240	240	240	240	240
p5.2.g	320	320	320	320	320	320	320	320	320	320	320	320
p5.2.h	404.5	410	410	410	410	410	410	410	410	410	410	410
p5.2.i	480	480	480	480	480	480	480	480	480	480	480	480
p5.2.j	580	580	580	580	580	580	580	580	580	580	580	580
p5.2.k	670	670	670	670	670	670	670	670	670	670	670	670
p5.2.l	778	800	800	800	800	800	800	800	800	800	800	–
p5.2.m	859.5	860	860	860	855	860	860	860	860	860	860	–
p5.2.n	921	925	925	925	920	925	925	925	925	925	925	–
p5.2.o	1011	1020	1020	1020	1010	1020	1020	1020	1020	1020	1020	–
p5.2.p	1143.5	1150	1150	1150	1140	1150	1150	1150	1150	1150	1150	–
p5.2.q	1194	1195	1195	1195	1185	1190	1195	1195	1195	1195	1195	–
p5.2.r	1258.5	1260	1260	1260	1250	1255	1260	1260	1260	1260	1260	–
p5.2.s	1324	1340	1340	1340	1300	1315	1325	1326.7	1330	1340	1340	–
p5.2.t	1382	1400	1400	1400	1360	1380	1390	1396.7	1400	1400	1400	–
p5.2.u	1452.5	1460	1460	1460	1450	1450	1455	1458.3	1460	1460	1460	–
p5.2.v	1491.5	1505	1505	1505	1480	1490	1500	1501.7	1505	1505	1505	–
p5.2.w	1537.5	1560	1560	1565	1560	1560	1560	1560	1560	1565	1565	–
p5.2.x	1595.5	1610	1595	1610	1590	1600	1610	1610	1610	1610	1610	–
p5.2.y	1631.5	1645	1635	1635	1620	1635	1645	1645	1645	1645	1645	–
p5.2.z	1672.5	1680	1670	1670	1680	1680	1680	1680	1680	1680	1680	–
p5.3.b	15	15	15	15	15	15	15	15	15	15	15	15
p5.3.c	20	20	20	20	20	20	20	20	20	20	20	20
p5.3.d	60	60	60	60	60	60	60	60	60	60	60	60
p5.3.f	110	110	110	110	110	110	110	110	110	110	110	110
p5.3.g	185	185	185	185	185	185	185	185	185	185	185	185
p5.3.h	260	260	260	260	260	260	260	260	260	260	260	260
p5.3.i	335	335	335	335	335	335	335	335	335	335	335	335
p5.3.j	470	470	470	470	470	470	470	470	470	470	470	470
p5.3.k	495	495	495	495	495	495	495	495	495	495	495	495
p5.3.l	590	595	595	595	585	595	595	595	595	595	595	595
p5.3.m	649.5	650	650	650	650	650	650	650	650	650	650	650
p5.3.n	755	755	755	755	755	755	755	755	755	755	755	755
p5.3.o	865	870	870	870	870	870	870	870	870	870	870	870
p5.3.p	990	990	990	990	990	990	990	990	990	990	990	990

Table 5 continued

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p5.3.q$	1061.5	1070	1070	1070	1065	1070	1070	1070	1070	1070	1070	–
$p5.3.r$	1114.5	1125	1125	1125	1120	1125	1125	1125	1125	1125	1125	–
$p5.3.s$	1187	1190	1190	1190	1185	1185	1190	1190	1190	1190	1190	–
$p5.3.t$	1251	1260	1260	1260	1250	1260	1260	1260	1260	1260	1260	–
$p5.3.u$	1336	1345	1345	1345	1325	1345	1345	1345	1345	1345	1345	–
$p5.3.v$	1402	1425	1425	1425	1410	1420	1425	1425	1425	1425	1425	–
$p5.3.w$	1458	1485	1485	1485	1455	1475	1485	1485	1485	1485	1485	–
$p5.3.x$	1513.5	1540	1555	1555	1520	1530	1530	1545	1555	1555	1555	–
$p5.3.y$	1555	1590	1595	1595	1570	1580	1590	1590	1590	1595	1595	–
$p5.3.z$	1610	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	–
$p5.4.c$	20	20	20	20	20	20	20	20	20	20	20	20
$p5.4.d$	20	20	20	20	20	20	20	20	20	20	20	20
$p5.4.e$	20	20	20	20	20	20	20	20	20	20	20	20
$p5.4.f$	80	80	80	80	80	80	80	80	80	80	80	80
$p5.4.g$	140	140	140	140	140	140	140	140	140	140	140	140
$p5.4.h$	140	140	140	140	140	140	140	140	140	140	140	140
$p5.4.i$	240	240	240	240	240	240	240	240	240	240	240	240
$p5.4.j$	340	340	340	340	340	340	340	340	340	340	340	340
$p5.4.k$	340	340	340	340	340	340	340	340	340	340	340	340
$p5.4.l$	429.5	430	430	430	430	430	430	430	430	430	430	430
$p5.4.m$	554	555	555	555	550	555	555	555	555	555	555	555
$p5.4.n$	620	620	620	620	620	620	620	620	620	620	620	620
$p5.4.o$	690	690	690	690	680	690	690	690	690	690	690	690
$p5.4.p$	758	765	765	765	760	760	760	760	760	765	765	765
$p5.4.q$	851	860	860	860	860	860	860	860	860	860	860	860
$p5.4.r$	960	960	960	960	940	960	960	960	960	960	960	960
$p5.4.s$	1020	1030	1030	1030	1025	1025	1025	1026.7	1030	1030	1030	–
$p5.4.t$	1152	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160
$p5.4.u$	1300	1300	1300	1300	1280	1300	1300	1300	1300	1300	1300	1300
$p5.4.v$	1320	1320	1320	1320	1310	1320	1320	1320	1320	1320	1320	1320
$p5.4.w$	1373.5	1390	1390	1390	1375	1380	1380	1380	1380	1390	1390	–
$p5.4.x$	1443	1450	1450	1450	1440	1445	1440	1445	1450	1450	1450	–
$p5.4.y$	1513	1520	1520	1520	1510	1510	1520	1520	1520	1520	1520	–
$p5.4.z$	1585.5	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	–

and so on. The procedure stops after n^2 Destruction/Construction iterations without improvement. After n iterations without improvement we apply the diversification components. This involves destroying a large part of the solution while removing a number, bounded by n/m rather than by 3, of customers from tours then applying

Table 6 Detailed results of different methods on benchmark instances set number 6 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
p6.2.d	189	192	192	192	192	192	192	192	192	192	192	192
p6.2.e	359.4	360	360	360	360	360	360	360	360	360	360	360
p6.2.f	587.4	588	588	588	588	588	588	588	588	588	588	588
p6.2.g	660	660	660	660	660	660	660	660	660	660	660	660
p6.2.h	780	780	780	780	780	780	780	780	780	780	780	780
p6.2.i	888	888	888	888	888	888	888	888	888	888	888	888
p6.2.j	947.4	948	948	948	930	936	942	944	948	948	948	–
p6.2.k	1032	1032	1032	1032	1026	1026	1032	1032	1032	1032	1032	–
p6.2.l	1111.2	1116	1116	1116	1086	1104	1116	1116	1116	1116	1116	–
p6.2.m	1184.4	1188	1188	1188	1176	1176	1188	1188	1188	1188	1188	–
p6.2.n	1230.6	1260	1242	1260	1230	1260	1254	1258	1260	1260	1260	–
p6.3.g	278.4	282	282	282	282	282	282	282	282	282	282	282
p6.3.h	427.8	444	444	444	444	444	444	444	444	444	444	444
p6.3.i	640.8	642	642	642	642	642	642	642	642	642	642	642
p6.3.j	825.6	828	828	828	828	828	828	828	828	828	828	828
p6.3.k	888.6	894	894	894	888	894	894	894	894	894	894	894
p6.3.l	996	1002	1002	1002	990	996	1002	1002	1002	1002	1002	1002
p6.3.m	1071.6	1080	1080	1080	1068	1080	1080	1080	1080	1080	1080	–
p6.3.n	1159.2	1170	1170	1170	1158	1164	1170	1170	1170	1170	1170	1170
p6.4.l	671.4	696	696	696	684	696	696	696	696	696	696	696
p6.4.m	885.6	912	912	912	912	912	912	912	912	912	912	912
p6.4.n	1061.4	1068	1068	1068	1068	1068	1068	1068	1068	1068	1068	1068

2-opt to each tour to optimize the travel cost, and finally performing the reconstruction phase.

2.4 Memetic algorithm

The algorithm starts with an initialization in which a small part of the population is created with an IDCH heuristic and the remainder is generated randomly. At each iteration a couple of parents is chosen among the population using the Binary Tournament (Prins 2004), which showed to be more efficient than random selection and the Roulette–Wheel procedure. The LOX crossover (Prins 2004) operator is used to produce a child chromosome. New chromosomes are evaluated using the Optimal Splitting procedure described in the previous section. They are then inserted into the current population using a simple and fast insertion technique to maintain a population of constant size, avoiding redundancy between chromosomes. The population is a list of chromosomes sorted lexicographically with respect to two criteria: the profit associated with the chromosome and the total travel cost. If a chromosome with the

Table 7 Detailed results of different methods on benchmark instances set number 7 for the TOP

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
*p*7.2.*a*	30	30	30	30	30	30	30	30	30	30	30	30
*p*7.2.*b*	64	64	64	64	64	64	64	64	64	64	64	64
*p*7.2.*c*	101	101	101	101	101	101	101	101	101	101	101	101
*p*7.2.*d*	190	190	190	190	190	190	190	190	190	190	190	190
*p*7.2.*e*	290	290	290	290	290	290	290	290	290	290	290	290
*p*7.2.*f*	386.7	387	387	387	384	384	384	385	387	387	387	387
*p*7.2.*g*	459	459	459	459	459	459	459	459	459	459	459	–
*p*7.2.*h*	521	521	516	521	516	517	521	521	521	521	521	–
*p*7.2.*i*	578.6	580	579	579	567	576	578	578.3	579	580	580	–
*p*7.2.*j*	644	646	641	644	630	641	644	645.3	646	646	646	–
*p*7.2.*k*	701.2	705	704	705	702	704	704	704	704	705	705	–
*p*7.2.*l*	765.4	767	767	767	751	757	767	767	767	767	767	–
*p*7.2.*m*	827	827	827	827	805	819	827	827	827	827	827	–
*p*7.2.*n*	878	888	888	888	849	871	884	885.3	888	888	888	–
*p*7.2.*o*	940.1	945	945	945	909	929	945	945	945	945	945	–
*p*7.2.*p*	991.3	1002	1002	1002	952	964	1002	1002	1002	1002	1002	–
*p*7.2.*q*	1040	1043	1042	1043	993	1017	1042	1043.3	1044	1044	1044	–
*p*7.2.*r*	1078.9	1094	1089	1094	1037	1051	1094	1094	1094	1094	1094	–
*p*7.2.*s*	1115.2	1136	1121	1135	1095	1120	1136	1136	1136	1136	1136	–
*p*7.2.*t*	1146.6	1179	1163	1179	1125	1149	1170	1176	1179	1179	1179	–
*p*7.3.*b*	46	46	46	46	46	46	46	46	46	46	46	46
*p*7.3.*c*	79	79	79	79	79	79	79	79	79	79	79	79
*p*7.3.*d*	117	117	117	117	117	117	117	117	117	117	117	117
*p*7.3.*e*	175	175	175	175	170	175	175	175	175	175	175	175
*p*7.3.*f*	247	247	247	247	247	247	247	247	247	247	247	247
*p*7.3.*g*	344	344	344	344	344	344	344	344	344	344	344	344
*p*7.3.*h*	424.3	425	425	425	419	425	425	425	425	425	425	425
*p*7.3.*i*	485.3	487	487	487	479	480	485	486.3	487	487	487	487
*p*7.3.*j*	563.2	564	562	564	556	558	563	563	563	564	564	–
*p*7.3.*k*	629.5	633	618	633	608	619	630	632	633	633	633	–
*p*7.3.*l*	680.7	684	682	683	675	681	681	681.7	683	684	684	–
*p*7.3.*m*	759.1	762	745	762	745	749	762	762	762	762	762	–
*p*7.3.*n*	813.9	820	813	813	790	800	814	818	820	820	820	–
*p*7.3.*o*	874	874	874	874	851	854	859	864	874	874	874	–
*p*7.3.*p*	925.6	929	925	927	900	919	923	925.7	927	929	929	–
*p*7.3.*q*	984.5	987	968	987	954	964	987	987	987	987	987	–
*p*7.3.*r*	1018.4	1026	1022	1026	1003	1020	1020	1022.7	1024	1026	1026	–
*p*7.3.*s*	1070.3	1081	1074	1081	1038	1054	1073	1078.3	1081	1081	1081	–
*p*7.3.*t*	1107.2	1118	1111	1117	1075	1086	1120	1120	1120	1118	1120	–

Table 7 continued

File	ACO_{Seq}		VNS_{Slow}		POP_b		MA			PBest	Best	UB
	$\bar{z}$	z_b	z_w	z_b	z_w	z_b	z_w	$\bar{z}$	z_b			
$p7.4.b$	30	30	30	30	30	30	30	30	30	30	30	30
$p7.4.c$	46	46	46	46	46	46	46	46	46	46	46	46
$p7.4.d$	79	79	79	79	79	79	79	79	79	79	79	79
$p7.4.e$	123	123	123	123	123	123	123	123	123	123	123	123
$p7.4.f$	164	164	164	164	164	164	164	164	164	164	164	164
$p7.4.g$	217	217	217	217	217	217	217	217	217	217	217	217
$p7.4.h$	285	285	285	285	283	285	285	285	285	285	285	285
$p7.4.i$	366	366	366	366	366	366	366	366	366	366	366	366
$p7.4.j$	462	462	462	462	462	462	462	462	462	462	462	462
$p7.4.k$	518	520	518	520	511	514	518	518	518	520	520	520
$p7.4.l$	581.7	590	590	590	576	586	586	587.3	590	590	590	590
$p7.4.m$	643.9	646	645	646	645	646	646	646	646	646	646	–
$p7.4.n$	725.6	730	725	726	699	725	725	725.7	726	730	730	–
$p7.4.o$	777.5	781	777	781	759	774	778	778.7	779	781	781	–
$p7.4.p$	839.4	846	844	846	818	824	844	845.3	846	846	846	–
$p7.4.q$	905.1	909	903	909	887	902	905	906.3	907	909	909	–
$p7.4.r$	969.2	970	969	970	953	966	970	970	970	970	970	–
$p7.4.s$	1017.7	1022	1021	1022	974	1001	1019	1021	1022	1022	1022	–
$p7.4.t$	1072.8	1077	1077	1077	1034	1042	1077	1077	1077	1077	1077	–

Table 8 Overall performance of each algorithm

	$\Delta Z_{\min}$	$\Delta Z_{\max}$	ΔZ
CGW	4340	N/A	N/A
TMH	2404	N/A	N/A
$\text{TS}_{\text{Penalty}}$	2376	981	1395
$\text{TS}_{\text{Feasible}}$	1184	399	785
VNS_{Fast}	1436	352	1084
VNS_{Slow}	427	84	343
KCS	N/A	223	N/A
ACO_{Seq}	N/A	204	N/A
ACO_{DC}	N/A	692	N/A
ACO_{RC}	N/A	775	N/A
ACO_{Sim}	N/A	659	N/A
IDCH	2979	1402	1577
MA	434	80	354

same profit and the same travel cost exists in the population, it is replaced with the new one. Otherwise, the chromosome is inserted and the worst chromosome of the new population is deleted. A child chromosome has a probability *pm* of being mutated, using a set of LS techniques repeatedly while improving. The stop condition of the MA is a bound on the number of iterations without improvement of the population, that is to say the number of iterations where the child chromosome simply replaces an existing chromosome in the population, or where its evaluation is worse than the worst chromosome in the current population. At the end of the search the chromosome at the head of the population is reported as the best solution. So the memetic can be resumed in Algorithm 1.

```
Data:
POP: population of solution;
N: population size;
Result:
S_POP[N-1], best solution found;
begin
    initialize POP with IDCH(see Sect. 2.3);
    fill POP with randomly generated solutions;
    keep POP ordered and update N;
    while NOT(stopping condition) do
        select 2 parents POP[p1] and POP[p2] using binary tournament;
        C ← LOX(POP[p1], POP[p2]);
        if (mutation) then
            C ← LocalSearch(C) (see Sect. 2.2);
        end
        if (P(S_C) ≥ P(S_POP[0])) then
            if (∄p| P(S_POP[p]) = P(S_C)) then
                eject POP[0] from POP;
                reset stopping condition;
            else
                update stopping condition;
            end
            insert or replace C in right place in POP;
        else
            update stopping condition;
        end
    end
end
```

Algorithm 1: Memetic algorithm applied to the TOP

3 Numerical results

We tested our MA on standard instances for the TOP from Chao et al. (1996). Instances comprise 7 sets containing different numbers of customers. Inside each set customer positions are constant, but the number of vehicles m varies between 2 and 4, and the maximum tour duration L also varies so that the number of customers that can really be serviced is different for each instance.

We set parameter values for our algorithm from a large number of experiments on these benchmark instances. The population size is fixed to 40 individuals. When the population is initialized, five individuals are generated by IDCH. Other individuals are generated randomly. The mutation rate pm of the MA is calculated as: $pm = 1 - \frac{\text{iter}_{\text{ineffective}}}{\text{iter}_{\max}}$. The algorithm stops when $\text{iter}_{\text{ineffective}}$, the number of elapsed consecutive iterations without improvement of the population, reaches $\text{iter}_{\max} = k \cdot n/m$ with $k = 5$.

For each instance, results are compared to those reported by Chao et al. (1996); Tang and Miller-Hooks (2005) and by Archetti et al. (2006). These results, as well as benchmark instances, are available at the URL: http://www-c.eco.unibs.it/~archetti/TOP.zip. We also compare our results to those reported by Khemakhem et al. (2007) and by Ke et al. (2008).

Archetti et al. (2006) proposed different methods: two TS and two VNS. For each method, they reported, for each instance, the worst profit z_w and the best profit z_b obtained from three executions. The difference $\Delta z = z_b - z_w$ is presented as an indicator of the stability of each method. Other results, Chao et al. (1996) and Tang and Miller-Hooks (2005), are given for a single execution. Khemakhem et al. (2007) proposed an algorithm based on adaptative memory, whose results are also presented for single execution since the method is fully deterministic. Ke et al. (2008) proposed four ACO methods whose tour construction procedures are different. They reported, for each instance and for each method, the best profit z_b and the average profit $\bar{z}$. In order that our method may be measured against the algorithms presented by Archetti et al. (2006) and by Ke et al. (2008), all of which have been shown to be very efficient, we report results of the MA the same way: we consider z_w, $\bar{z}$ and z_b for three executions of the MA. We also report the profit of the best solution of the initial population, POP_b, the same way in order to evaluate the efficiency of IDCH.

Tables 1, 2, 3, 4, 5, 6 and 7 report detailed results for each data set. Due to lack of space, we present only two methods showed to be the most efficient by their authors to compare with our MA and IDCH. Column headers are as follows: ACO_{Seq} denotes the ACO with sequential tour construction of Ke et al. (2008) and VNS_{Slow} is the most efficient method proposed by Archetti et al. (2006). The column PBest shows previous best results considered by other authors and the column Best summarizes the best results through all methods. The column *UB* corresponds to the upper bound of the profit obtained with an exact algorithm, if known. As far as we know, the only existing upper bound for the TOP is that described by Boussier et al. (2007). Some instances are such that L is so small that no customer can be serviced, they are trivial and so not reported on the tables.

As a synthesis of these results, we consider the sum of the differences between the best known value of the profit and z_b (resp. z_w) for each instance: $\Delta^{z_b}_{\text{Best}} = \text{Best} - z_b$ (resp. $\Delta^{z_w}_{\text{Best}}$). The *Best* value we consider is the best known profit of an instance, including our results.

Table 8 reports $\Delta Z_{\max} = \sum \Delta^{z_{\max}}_{\text{Best}}$ and $\Delta Z_{\min} = \sum \Delta^{z_{\min}}_{\text{Best}}$ for each method. The difference $\Delta Z = \Delta Z_{\max} - \Delta Z_{\min}$ between these two values is also given as an indicator of the stability of each method. Detailed results are also available at: http://www.hds.utc.fr/~boulyher/TOP/top.html. Our MA produced solutions that improve on the best known solutions from the literature for 5 instances of the benchmark set. Profits 1267, 1292, 1124, 1216 and 1120 have, respectively, been reached for instances $p4.2.q$, $p4.2.r$, $p4.4.p$, $p4.4.r$ and $p7.3.t$.

A comparison of profits with the upper bound of Boussier et al. (2007) shows that profits reached by CGW, $\text{TS}_{\text{Penalty}}$ and VNS_{Slow} exceed the upper bound on a subset of 8 instances. There is something abnormal about these results. Consequently these instances are not included in the results of Tables 1, 2, 3, 4, 5, 6 and 7, and details

Table 9 Results for instances for which some profits exceed the upper bound

	$p1.3.h$	$p1.3.o$	$p1.3.r$	$p2.3.h$	$p3.4.k$	$p5.3.e$	$p6.4.j$	$p6.4.k$
CGW	**75**	**215**	250	165	350	**110**	366	**546**
TMH	70	205	250	165	350	95	366	522
$\text{TS}_{\text{Penalty}}$								
z_w	70	205	250	165	350	95	366	528
z_b	70	205	250	**170**	350	95	366	528
$\text{TS}_{\text{Feasible}}$								
z_w	70	205	250	165	350	95	366	528
z_b	70	205	250	165	350	95	366	528
VNS_{Fast}								
z_w	70	205	250	165	350	95	366	528
z_b	70	205	250	165	350	95	366	528
VNS_{Slow}								
z_w	70	205	250	165	350	95	366	528
z_b	70	205	250	165	**370**	95	**390**	528
KCS	70	205	**255**	165	350	95	366	528
ACO								
z_b	70	205	250	165	350	95	366	528
z_w	70	205	250	165	350	95	366	528
z_b	70	205	250	165	350	95	366	528
Best	70	205	250	165	350	95	366	528
UB	70	205	250	165	350	95	366	528

about these instances are given in Table 9. Bold values identify profits that exceed the upper bound of Boussier et al. (2007).

Table 10 finally reports CPU time for each method and for each instance set from 1 to 7. We denote *cpu* the CPU time if a single execution was performed and *avg* and *max*, respectively, the average and the maximal CPU time if three executions were performed. Computers used for experiments are as follows:

– CGW: run on a SUN 4/730 Workstation,
– TMH: run on a DEC Alpha XP1000 computer,
– $\text{TS}_{\text{Penalty}}$, $\text{TS}_{\text{Feasible}}$, VNS_{Fast} and VNS_{Slow}: run on an Intel Pentium 4 personal computer with 2.8 GHz and 1048 MB RAM,
– KCS: run on an Intel Pentium 4 with 3.0 GHz and 512 MB RAM,
– ACO: run on an Intel personal computer with 3.0 GHz
– MA: run on a Intel Core 2 Duo E6750—2.67 GHz (no parallelization of the program) with 2 GB RAM.

These results clearly show our MA compares very well with state of the art methods. MA outperforms the VNS Slow algorithm of Archetti et al. (2006) in term of efficiency and is quite equivalent in term of stability. A comparison of computational

Table 10 Average and maximal CPU times for the different instance sets

Data set	1	2	3	4	5	6	7
CGW							
CPU	15.41	0.85	15.37	934.8	193.7	150.1	841.4
TMH							
CPU	N/A	N/A	N/A	796.7	71.3	45.7	432.6
$TS_{Penalty}$							
Avg	4.67	0.00	6.03	105.29	69.45	66.29	158.97
Max	10.00	0.00	10.00	612.00	147.00	96.00	582.00
$TS_{Feasible}$							
Avg	1.63	0.00	1.59	282.92	26.55	20.19	256.76
Max	5.00	0.00	9.00	324.00	105.00	48.00	514.00
VNS_{Fast}							
Avg	0.13	0.00	0.15	22.52	34.17	8.74	10.34
Max	1.00	0.00	1.00	121.00	30.00	20.00	90.00
VNS_{Slow}							
Avg	7.78	0.03	10.19	457.89	158.93	147.88	309.87
Max	22.00	1.00	19.00	1118.00	394.00	310.00	911.00
KCS							
Avg	7.20	2.90	8.30	130.70	37.10	46.80	69.20
Max	24.10	5.70	23.50	329.60	91.00	97.10	197.10
ACO							
Max	7.90	3.80	8.50	51.10	25.20	20.30	44.70
MA							
Avg	1.31	0.13	1.56	125.26	23.96	15.53	90.30
Max	4.11	0.53	3.96	357.05	80.19	64.29	268.01

times using similar computers shows, however, that MA outperforms VNS Slow on this point.

Conclusion

We propose a new resolution method for the TOP using the recent MA approach. It is the first time that an Evolutionary Algorithm has been used for this problem. We also propose an Optimal Split procedure as a key feature of this method especially intended for the TOP. Our method proved very efficient and fast compared with the best existing methods, and even produced improved solutions for some instances of the standard benchmark for the TOP.

These results show, first, that population-based algorithms can efficiently be applied to the TOP, as shown recently with ACO (Ke et al. 2008). Secondly, the use of the Optimal Splitting procedure shows that further research into specialized methods is a promising direction in addressing the TOP. Therefore, these results confirm that the association of optimal split with MAs is very fruitful for VRPs, as shown in anterior researches on the CARP by Lacomme et al. (2004) and the CVRP by Belenguer et al. (2006).

Acknowledgments The authors would like to thank two anonymous referees for their comments and suggestions that helped improving the quality of this paper.

References

Archetti C, Hertz A, Speranza M (2006) Metaheuristics for the team orienteering problem. J Heuristics 13(1):49–76

Beasley JE (1983) Route-first cluster-second methods for vehicle routing. Omega 11:403–408

Belenguer J-M, Benavent E, Lacomme P, Prins C (2006) Lower and upper bounds for the mixed capacitated arc routing problem. Comput Oper Res 33(12):3363–3383

Bellman R (1957) Dynamic Programming. Princeton University Press, Princeton

Bouly H, Dang D-C, Moukrim A (2008) A memetic algorithm for the team orienteering problem. In EvoWorkshops

Boussier S, Feillet D, Gendreau M (2007) An exact algorithm for team orienteering problems. 4OR 5(3):211–230

Butt S, Cavalier T (1994) A heuristic for the multiple tour maximum collection problem. Comput Oper Res 21:101–111

Chao I-M, Golden B, Wasil E (1996) The team orienteering problem. Eur J Oper Res 88:464–474

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2): 188–205

Ke L, Archetti C, Feng Z (2008) Ants can solve the team orienteering problem. Comput Ind Eng 54(3): 648–665, ISSN 0360-8352

Khemakhem M, Chabchoub H, Semet F (2007) Heuristique basée sur la mémoire adaptative pour le problème de tournées de véhicules sélectives. In Logistique & Transport, pp 31–37, Sousse, Tunisie

Lacomme P, Prins C, Ramdane-Cherif W (2004) Competitive memetic algorithms for arc routing problems. Ann Oper Res 131(1–4):159–185

Moscato P (1999) New ideas in optimization. chapter Memetic Algorithms: a short introduction. McGraw-Hill, UK, pp 219–234

Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 31(12):1985–2002

Ruiz R Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. EJOR 177:2033–2049

Solomon M (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35:254–265

Tang H, Miller-Hooks E (2005) A tabu search heuristic for the team orienteering problem. Comput Oper Res 32:1379–1407

Ulusoy G (1985) The fleet size and mixed problem for capacitated arc routing. Eur J Oper Res 22:329–337