A Memetic Algorithm for the Team Orienteering Problem

Hermann Bouly[1,2], Duc-Cuong Dang[1], and Aziz Moukrim[1]

[1] Université de Technologie de Compiègne
Heudiasyc, CNRS UMR 6599, BP 20529, 60205 Compiègne, France
[2] VEOLIA Environnement, Direction de la Recherche
17/19, rue La Pérouse, 75016 Paris, France
{hermann.bouly,duc-cuong.dang,aziz.moukrim}@hds.utc.fr

Abstract. The Team Orienteering Problem (TOP) is a generalization of the Orienteering Problem (OP). 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 (GA) 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 (MA) 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 11 instances.

1 Introduction

The Team Orienteering Problem first appeared in Butt and Cavalier [4] under the name of the Multiple Tour Maximum Collection Problem. The term TOP, first introduced in Chao, Golden and Wasil [5], 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, Golden and Wasil [5] 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

M. Giacobini et al. (Eds.): EvoWorkshops 2008, LNCS 4974, pp. 649–658, 2008.

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.

Recently Feillet, Dejax and Gendreau [6] 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. As far as we know, the most recent paper dealing with solution methods for the TOP is [7]. Most of the metaheuristics shown to be effective for the TOP are Tabu Search (TS) and Variable Neighborhood Search (VNS) [1,12]. The Memetic Algorithm (MA), first introduced by Moscato [8], is a recent technique that has been shown to be competitive for VRPs [9]. 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). 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 [10], combined with a priority rule and LS. Computational results are compared with those of Chao, Golden and Wasil [5], Tang and Miller-Hooks [12] and Archetti, Hertz and Speranza [1].

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 Section 3. At the end we put forward some conclusions.

2 Problem Formulation

The TOP can be modeled with a graph $G = (V, E)$, where $V = \{1, 2, ..., n\}$ 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. 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 limit L. So 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.

3 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 [2] and Ulusoy [13], 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.

A Memetic Algorithm (MA) is a combination of an Evolutionary Algorithm and Local Search techniques. This combination has been shown to be effective for the VRP in Prins [9]. Our MA is a combination of GA and some LS techniques.

3.1 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 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.

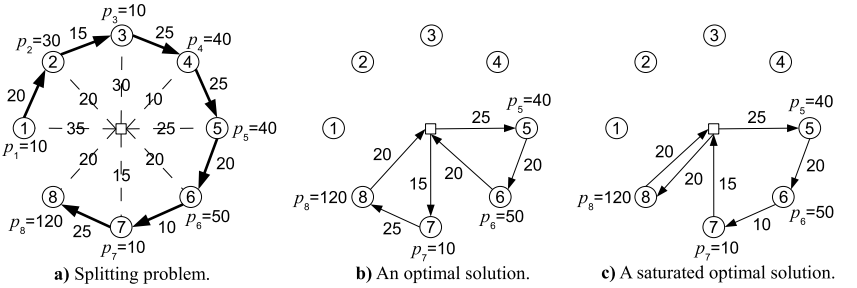

Fig. 1. A giant tour with 8 customers and two optimal solutions for $L = 70$

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 8 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 solutions, one of which is saturated, are shown in Figure 1.

The splitting problem consists in identifying a π-solution that maximizes the collected profit. Given these notations, we make the proof (that cannot be placed here because of space restriction) that an optimal splitting of the giant tour is obtained through consideration of only the saturated tours.

Proposition 1. *Let π be an arbitrary sequence of the vertices of an instance of the TOP. Then there exists a saturated optimal π-solution.*

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

Optimal Evaluation. The splitting problem can be formulated as finding a path on an acyclic graph $H = (X, F)$, where $X = \{d, 1, 1', 2, 2', ...n, n', a\}$. 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 $w_{x',y} = 0$. The graph construction is such that any path in the graph from departure to arrival nodes is $2q + 1$ arcs long and contains q compatible tours. Figure 2 shows the graph corresponding to the splitting problem in Figure 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.

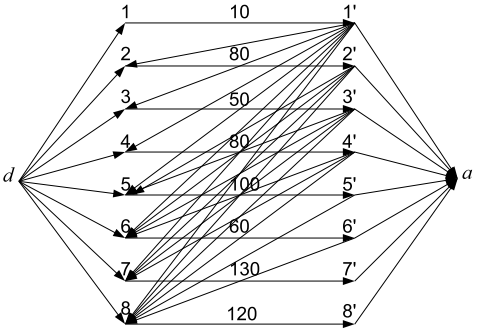

Fig. 2. Graph representation for the splitting problem

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 $1, 1', 2, 2', ..., n, n', a$. 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, the greatest value of μ_a indicates the final node of the longest path. 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 $O(m \cdot n^2)$.

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 enable us to obtain an approximate evaluation where the first tour begins with $i_1 = 1$. The complexity of this method is $O(n)$.

3.2 Local Search as Mutation Operator

To complete the memetic algorithm, local search techniques are used as mutation operators with probability pm. We use different neighborhoods during mutation. The procedure selects these neighborhoods in random order. Once an improvement is found, the search within the current neighborhood is stopped,

and we restart the procedure choosing a new neighborhood, until no further improvements are found.

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

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

In TSP problems, each neighbor obtained using the two operators described above is evaluated before a movement is carried out, leading to $O(n^2)$. 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 local search techniques used in mutation, we decided to use Quick Split in association with these two operators. A *compressed* version of the current chromosome is produced at the beginning of these local search techniques to left-shift tours within the sequence so that solutions produced by the Shift operator and the Swap operator can be evaluated by the Quick Split quickly and efficiently. At the end of these local search techniques, when a better 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 Destruction/Construction, [10]). This local search 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 [11]. 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 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.

3.3 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 [11] 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 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, and so on. The procedure stops after n^2 Destruction/Construction iterations without improvement. After each n iterations without improvement we apply the diversification components. This involves destroying a large part of the solution, applying 2-opt to each tour to optimize the travel cost, and finally performing the reconstruction phase.

3.4 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, which showed more efficient than random selection and the Roulette-Wheel procedures. The LOX crossover [9] 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 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 Local Search techniques repeatedly while improving. The stop condition of the Memetic Algorithm 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.

4 Numerical Results

We tested our MA on standard instances for the TOP from Chao, Golden and Wasil [5]. 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: $pm = 1 - \frac{iter_{ineffective}}{iter_{max}}$. The algorithm stops when $iter_{ineffective}$, the number of elapsed consecutive iterations without improvement of the population, reaches $iter_{max} = k \cdot n/m$ with $k = 5$.

For each instance, results are compared to results reported by Chao, Golden and Wasil [5], Tang and Miller-Hooks [12] and by Archetti, Hertz and Speranza [1]. These results, as well as benchmark instances, are available at the following url: `http://www-c.eco.unibs.it/~archetti/TOP.zip`.

Archetti, Hertz and Speranza [1] proposed different methods: two TS and two VNS. For each method, they reported, for each instance, the worst profit z_{min} and the best profit z_{max} obtained from three executions. The difference $\Delta z = z_{max} - z_{min}$ is presented as an indicator of the stability of each method. Other results (Chao, Golden and Wasil [5] and Tang and Miller-Hooks[12]) are given for a single execution. In order that our method may be measured against the algorithms presented by Archetti, Hertz and Speranza [1], all of which have been shown to be very efficient, we report results of the MA the same way: we consider z_{min} and z_{max} for three executions of the MA.

Because of space limitations we only report the sum of the differences between the best known value of the profit and z_{max} (resp. z_{min}) for each instance: $\Delta_{Best}^{z_{max}} = Best - z_{max}$ (resp. $\Delta_{Best}^{z_{min}}$). The $Best$ value we consider is the best known profit of an instance, including our results. More detailed results are available at: `http://www.hds.utc.fr/~boulyher/TOP/top.html`.

Table 1 reports $\Delta Z_{max} = \sum \Delta_{Best}^{z_{max}}$ and $\Delta Z_{min} = \sum \Delta_{Best}^{z_{min}}$ 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.

Column headers are as follows: $TS_{Penalty}$, $TS_{Feasible}$, VNS_{Fast} and VNS_{Slow} are the four methods proposed by Archetti, Hertz and Speranza [1], TMH denotes the Tabu Search of Tang and Miller-Hooks [12] and column CGW denotes the heuristic of Chao, Golden and Wasil [5]. 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, Feillet and Gendreau [3].

Our Memetic Algorithm produced solutions that improve on the best known solutions from the literature for 11 instances of the benchmark set. Profits 965, 1242, 1267, 1292, 1304, 1252, 1124, 1216, 1645, 646 and 1120 have respectively been reached for instances $p4.2.j$, $p4.2.p$, $p4.2.q$, $p4.2.r$, $p4.2.s$, $p4.3.q$, $p4.4.p$, $p4.4.r$, $p5.2.y$, $p7.2.j$ and $p7.3.t$.

A comparison of profits with the upper bound of Boussier, Feillet and Gendreau [3] shows that profits reached by $TS_{Penalty}$, VNS_{Slow} and CGW exceed the upper bound on a subset of seven instances. It seems abnormal, and these instances are consequently not included in the results of Table 1, and details about

Table 1. Overall performance of each algorithm

	$TS_{Penalty}$	$TS_{Feasible}$	VNS_{Fast}	VNS_{Slow}	TMH	CGW	MA
ΔZ_{min}	2370	1178	1430	421	2398	4334	428
ΔZ_{max}	975	393	346	78	N/A	N/A	**74**
ΔZ	1395	785	1084	343	N/A	N/A	354

Table 2. Results for instances for which some profits exceed the upper bound

	$TS_{Penalty}$		$TS_{Feasible}$		VNS_{Fast}		VNS_{Slow}		TMH	CGW	MA		$Best$	UB
$file$	z_{min}	z_{max}	z_{min}	z_{max}	z_{min}	z_{max}	z_{min}	z_{max}			z_{min}	z_{max}		
$p1.3.h$	70	70	70	70	70	70	70	70	70	**75**	70	70	70	70
$p1.3.o$	205	205	205	205	205	205	205	205	205	**215**	205	205	205	205
$p2.3.h$	165	**170**	165	165	165	165	165	165	165	165	165	165	165	165
$p3.4.k$	350	350	350	350	350	350	350	**370**	350	350	350	350	350	350
$p5.3.e$	95	95	95	95	95	95	95	95	95	**110**	95	95	95	95
$p6.4.j$	366	366	366	366	366	366	366	**390**	366	366	366	366	366	366
$p6.4.k$	528	528	528	528	528	528	528	528	522	**546**	528	528	528	528

Table 3. Average and maximal CPU times for the different instance sets

	TMH	CGW	$TS_{Penalty}$		$TS_{Feasible}$		VNS_{Fast}		VNS_{Slow}		MA	
	cpu	cpu	avg	max	avg	max	avg	max	avg	max	avg	max
1	N/A	15.41	4.67	10.00	1.63	5.00	0.13	1.00	7.78	22.00	1.31	4.11
2	N/A	0.85	0.00	0.00	0.00	0.00	0.00	0.00	0.03	1.00	0.13	0.53
3	N/A	15.37	6.03	10.00	1.59	9.00	0.15	1.00	10.19	19.00	1.56	3.96
4	796.7	934.8	105.29	612.00	282.92	324.00	22.52	121.00	457.89	1118.00	125.26	357.05
5	71.3	193.7	69.45	147.00	26.55	105.00	34.17	30.00	158.93	394.00	23.96	80.19
6	45.7	150.1	66.29	96.00	20.19	48.00	8.74	20.00	147.88	310.00	15.53	64.29
7	432.6	841.4	158.97	582.00	256.76	514.00	10.34	90.00	309.87	911.00	90.30	268.01

these instances are given in Table 2. Bold values identify profits that exceed the upper bound of Boussier, Feillet and Gendreau [3].

Table 3 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,
- $TS_{Penalty}$,$TS_{Feasible}$,VNS_{Fast} and VNS_{Slow}: run on an Intel Pentium 4 personal computer with 2.8 GHz and 1048 MB RAM,
- 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 Memetic Algorithm compares very well with state of the art methods. MA outperforms the VNS Slow algorithm of Archetti,

Hertz and Speranza in term of efficiency and is quite equivalent in term of stability. A comparison of computational times using similar computers shows, however, that MA outperforms VNS Slow on this point.

5 Conclusion

We propose a new resolution method for the TOP using the recent Memetic Algorithm 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 Team Orienteering Problem. Secondly, the use of the Optimal Splitting procedure shows that further research into specialized methods is a promising direction in addressing the Team Orienteering Problem.

References

1. Archetti, C., Hertz, A., Speranza, M.G.: Metaheuristics for the team orienteering problem. Journal of Heuristics 13(1), 49–76 (2006)
2. Beasley, J.E.: Route-first cluster-second methods for vehicle routing. Omega 11, 403–408 (1983)
3. Boussier, S., Feillet, D., Gendreau, M.: An exact algorithm for the team orienteering problems. 4OR 5, 211–230 (2007)
4. Butt, S., Cavalier, T.: A heuristic for the multiple tour maximum collection problem. Computers & Operations Research 21, 101–111 (1994)
5. Chao, I.-M., Golden, B., Wasil, E.A.: The team orienteering problem. European Journal of Operational Research 88, 464–474 (1996)
6. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science 39(2), 188–205 (2005)
7. Khemakhem, M., Chabchoub, H., Semet, F.: Heuristique basée sur la mémoire adaptative pour le problème de tournées de véhicules sélectives. In: Logistique & Transport, Sousse, Tunisie, pp. 31–37 (November 2007)
8. Moscato, P.: New Ideas in Optimization, chapter Memetic Algorithms: a short introduction, pp. 219–234 (1999)
9. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computer & Operations Research 31(12), 1985–2002 (2004)
10. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. EJOR 177, 2033–2049 (2007)
11. Solomon, M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265 (1987)
12. Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computer & Operations Research 32, 1379–1407 (2005)
13. Ulusoy, G.: The fleet size and mixed problem for capacitated arc routing. European Journal of Operational Research 22, 329–337 (1985)