
GRASP with Path Relinking for the Orienteering Problem

VICENTE CAMPOS

Departamento de Estadística e Investigación Operativa
Universitat de València, Spain
vicente.campos@uv.es

RAFAEL MARTÍ

Departamento de Estadística e Investigación Operativa
Universitat de València, Spain
rafael.marti@uv.es

ABSTRACT

In this paper, we address an optimization problem resulting from the combination of the well-known traveling salesman and knapsack problems. In particular, we target the orienteering problem, originated in the context of sports, which consists of maximizing the total score associated with the vertices visited in a path within the available time. The problem, also known as the selective travelling salesman problem, is NP-hard and can be formulated as an integer linear program. Since the 1980s, several solution methods for this problem have been developed and applied to a variety of fields, particularly in routing and tourism. We propose a heuristic method –based on the GRASP and the path relinking methodologies – for finding approximate solutions to this optimization problem. We explore different constructive methods and combine two neighborhoods in the local search of GRASP. Our experimentation with 73 previously reported instances shows that the proposed procedure obtains high quality solutions employing short computing times.

Keywords: Metaheuristics, GRASP, Path Relinking, Orienteering problem.

Version: November 9, 2011

1. Introduction

The orienteering problem (OP), originated in the context of sports, consists of determining a path from an origin to a destination in a graph (directed or undirected), through a subset of locations in order to maximize the sum of the scores of the visited locations. Not all locations (vertices in the graph) can be visited since the available time (or total distance) is limited. Different applications can be found for this problem, for example in tourism we want to plan a tourist route in a large city, giving scores to the locations (in terms of their cultural interest for instance) and the tour visiting the selected vertices cannot exceed a maximum length or time (Tsiligirides, 1984).

Let $G(V, A)$ be a directed graph where V ($|V| = n$) and A ($|A| = m$) represent respectively the set of vertices and arcs. Each vertex $v_i \in V$ has a non-negative score S_i for $i = 1, \dots, n$, and each arc $(i, j) \in A$ has an associated travel time t_{ij} for $i, j = 1, \dots, n$. The OP consists of determining a path from v_1 to v_n visiting some of the vertices in V in a way that the total travel time does not exceed a pre-established limit T_{max} , maximizing the sum of the associated scores.

We can formulate the OP as a linear integer problem (Vansteenwegen et al., 2011) by defining the binary variables $x_{ij} = 1$ if vertex i is visited followed by vertex j , and $x_{ij} = 0$ otherwise, for $i, j = 1, \dots, n$. In this formulation, originally proposed by Miller et al. (1960) in the context of the TSP, we also need the integer variables u_i with the position of vertex i in the path, $i = 1, \dots, n$. In mathematical terms:

$$\text{Max} \quad \sum_{i=2}^{n-1} \sum_{j=2}^n S_i x_{ij}$$

s.t.:

$$\sum_{j=2}^n x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 \quad (1)$$

$$\sum_{j=2}^n x_{kj} = \sum_{i=1}^{n-1} x_{ik} \leq 1 \quad k = 2, \dots, n-1 \quad (2)$$

$$\sum_{i=1}^{n-1} \sum_{j=2}^n t_{ij} x_{ij} \leq T_{max} \quad (3)$$

$$2 \leq u_i \leq n \quad i = 2, \dots, n \quad (4)$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) \quad i, j = 2, \dots, n \quad (5)$$

$$x_{ij} \in \{0,1\} \quad 1 \leq i, j \leq n, \quad u_i \in \mathbb{Z} \quad i = 2, \dots, n$$

In the formulation above, constraints (1) force the path to start at vertex 1 and to end at vertex n , while constraints (2) guarantee that every vertex in the graph is visited at most once. The time limit constraint (3) and the sub-tour elimination constraints (4 and 5) complement the formulation.

We do not include a discussion of previous work on the orienteering problem because fairly complete reviews have appeared in recent publications, including Vansteenwegen et al. (2011). They clearly stated that the methods proposed in the most recent solution approach for the OP (Schilde et al. 2009), Ant colony optimization and VNS, improve upon all the previous methods, including the classic five-step heuristic of Chao et al. (1996). A Path Relinking approach was applied for the Team Orienteering Problem in Souffriau et al. 2010, in this variant of the OP, a set of routes have to maximize the total score of the visited vertices, each route cannot exceed a common limit time. On the other hand, we have also identified an exact branch-and-cut algorithm to optimally solve medium-sized instances (Fischetti et al. 1998), but it is very time consuming. For this reason many researchers have focused their interest in heuristics. Our main contribution is the development and testing parameters of a solution method for the OP based on the GRASP and Path Relinking methodologies (Resende and Ribeiro 2001) that compete with these previous heuristics reported in the literature.

2. Constructive Methods

In this section we propose four constructive methods for the orienteering problem based on GRASP (Resende and Ribeiro 2003).

Given a graph $G(V, A)$, constructive C1 starts with a path of length one in which we directly go from 1 to n through the arc $(1, n)$. The set $P = \{1, n\}$ represents the partial solution under construction and T_p its associated travel time (initially $T_p = t_{1n}$). At each step, C1 selects a candidate element $i \in CL$ to be included in the solution. The candidate list is formed with all the vertices not present in the path that can be added within the time limit. In the first iteration

$$CL = \{i \in V \setminus P : t_{1i} + t_{in} \leq T_{max}\}.$$

C1 implements a typical GRASP construction in which first each candidate element is evaluated by a greedy function to construct the restricted candidate list RCL and then an element is selected at random from the RCL . In particular, we compute the maximum score S_{max} of the elements in CL as

$$S_{max} = \max_{i \in CL} S_i,$$

then we construct RCL with all the candidate elements with a score value within a fraction α of the maximum score. In mathematical terms:

$$RCL = \{i \in CL : S_i \geq \alpha S_{max}\}.$$

Finally, C1 randomly selects an element in RCL and inserts it in the best position in the path P . C1 performs iterations as long as new vertices can be added to the partial path (i.e., as long as CL is not empty). When no element out of the path can be inserted in it within the time limit, C1 stops and returns the path as the solution.

The randomization in C1 permits running it several times, say Max_iter iterations. Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. Based on empirical observations, it has been found that the sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction (that one with the same GRASP elements but replacing the random selection with the best available one), but the best overall trial dominates the deterministic solution with a high probability. The intuitive justification of this phenomenon is based on the ordering statistics of

sampling. It implements a way of independently sampling the solution space and each construction consists of an independent algorithm. In this sense, GRASP is a memory-less method since no information is recorded from one construction to the next.

We now consider C2, an alternative construction procedure introduced in Resende and Werneck (2004) as random plus greedy construction. In C2 we first determine the number of candidates in the candidate list $RCL2$ as a fraction α of the elements in CL , ($|RCL2| = \lceil \alpha |CL| \rceil$). Then a random sample of size $|RCL2|$ is taken from CL and the element $i \in RCL2$ with the maximum score S_i is selected. As C1, C2 inserts the selected element in the best position in the path P . It performs iterations as long as new vertices can be added to the partial path.

Constructive method C3 also implements a typical GRASP construction, as C1, in which first each candidate element is evaluated by a greedy function to construct RCL , from which an element is randomly selected. The candidate set of elements CL is computed in the same way as in C1, but the greedy evaluation e_i of element i , consists now on the quotient between the score S_i and the increment in the time, Δt_i , when the element is inserted in the path (in the best position). In mathematical terms:

$$e_i = \frac{S_i}{\Delta t_i}$$

As it is customary in GRASP, we construct RCL with the elements in CL with an evaluation within a fraction α of the maximum value. In mathematical terms:

$$RCL = \{i \in CL : e_i \geq \alpha e_{max}\} \text{ where } e_{max} = \max_{i \in CL} e_i$$

Finally, C4 alternates in C3 the random and the greedy choices, as C2 does with C1. In particular, C4 first constructs the restricted candidate list $RCL2$ with a fraction α of the elements in CL selected at random ($|RCL2| = \lceil \alpha |CL| \rceil$). Then, it evaluates all the elements in $RCL2$, computing e_i for all $i \in RCL2$, and selects the best one, i.e. the element with maximum quotient between the score and the time increment. As the three previous methods, C4 inserts the selected element in the best position in the P path under construction.

2.1 Comparison of Methods

When we design a constructive method as a part of a larger solving method it has to be able to produce good starting points for the master method. This is especially true in the case of GRASP with Path Relinking (PR) in which we apply first the local search and then the PR to the constructed solutions. In this context, we want the constructive method to obtain good solutions in terms of the objective function, but also diverse to reach different regions of the solution space.

Considering the four constructive methods proposed above, a way of comparing them is to generate a set of solutions with each one and compare their quality and diversity. Since the quality is directly measured by the objective function, we now propose a measure of diversity. Given two solutions, $A = \{1, a_1, a_2, \dots, a_k, n\}$ and $B = \{1, b_1, b_2, \dots, b_t, n\}$ we compute their diversity, $div(A, B)$, as the number of elements (vertices in the graph) in A not present in B , plus the number of elements in B not present in A . To make this value independent of the size of the problem and of the maximum time limit T_{max} , we divide it by the maximum number of different elements that they may have (i.e., $k + t$).

To illustrate, suppose two solutions in a graph with $n = 21$, $A = \{1,7,10,4,8,3,12,20,21\}$ with 7 elements (without computing the origin 1 and destination 21) and $B = \{1,12,3,9,11,7,21\}$ with 5 elements. The number of elements in A not present in B is 4, and the number of elements in B not present in A is 2. Then, the diversity value between A and B is:

$$div(A, B) = \frac{4 + 2}{7 + 5} = 0.5$$

We then generate 100 solutions with each constructive method and compute the average diversity value between all pairs of solutions. On the other hand, we compute the objective function value of each solution and its associated relative deviation from the optimal solution value. The averages of these deviation and diversity values provide an overall evaluation of the method.

To test this point with our four constructive methods and the different values of their associated parameter, we consider 33 instances with optimum known in Fischetti et al. (1998). Table 1 shows for each method, C1, C2, C3 and C4, and each value of the parameter α tested, 0.2, 0.4, 0.6 and 0.8, the average across the 33 instances of the average deviation value (Dev) and the average diversity value (Div). This table also reports the number of best solutions that each method is able to match.

	α	0.2	0.4	0.6	0.8
C1	Dev	19.3%	13.5%	12.1%	13.0%
	Div	0.57	0.49	0.43	0.32
C2	Dev	7.2%	7.9%	8.4%	9.9%
	Div	0.28	0.20	0.15	0.11
C3	Dev	11.8%	13.1%	13.8%	14.2%
	Div	0.41	0.33	0.27	0.21
C4	Dev	7.4%	7.3%	7.9%	9.2%
	Div	0.25	0.18	0.14	0.11

Table 1. Average deviation from optimal values and diversity of constructive methods.

Table 1 shows that the best method in terms of quality is C2 (with $\alpha = 0.2$) since it is able to obtain a 7.2% deviation, which compares favorably with the rest of the methods. We also observe that the best method in terms of diversity is C1 with an average value of 0.57. Given their different nature and range, it is difficult to directly compare both measures (Div. and Dev.) in order to establish the best constructive method overall. Figure 1 depicts a point for each of the 16 methods reported in Table 1 (the four methods with the four values of α). The x-axis represents the diversity value (Div.) and the y-axis the deviation value in the range $[0,1]$ (i.e., we represent the 1-Dev. value). In this way, for both values in the diagram we have that the larger the value the better the method.

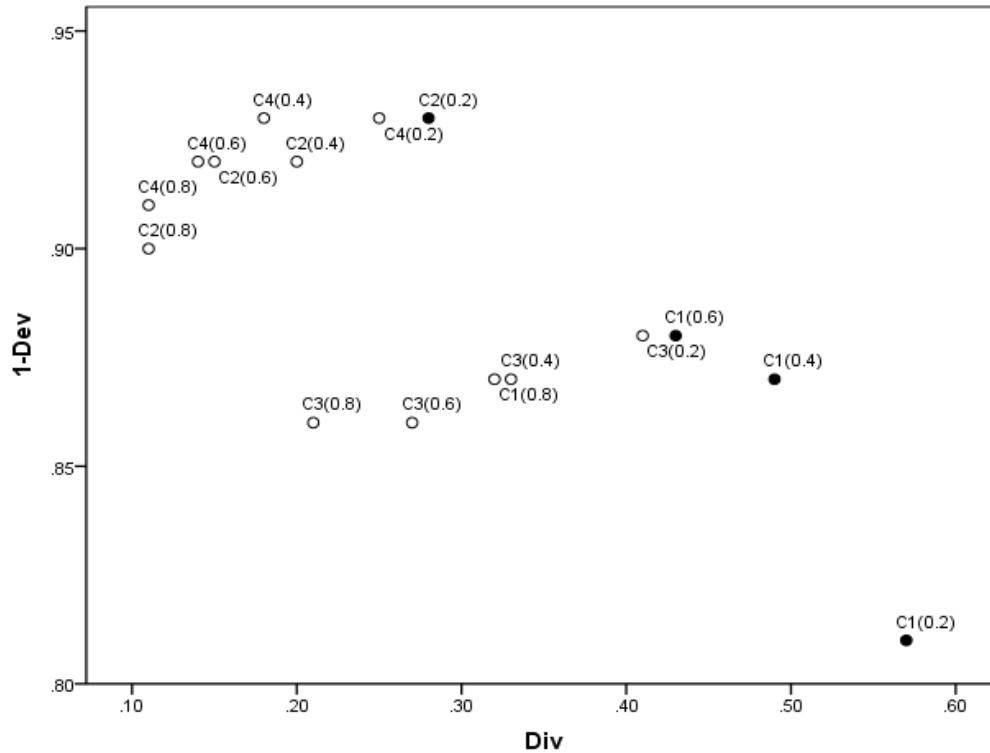


Figure 1. Quality and diversity of constructive methods.

If we analyze the points in Figure 1 from a bi-objective perspective, in which an objective is maximize the quality (1-Dev) and the other maximize diversity (Div), we conclude that there are only four non-dominated points (methods): C2(0.2), C1(0.6), C1(0.4) and C1(0.2). We say that a method is non-dominated if there is no other method with a better value in one objective and a better or equal value in the other.

It is difficult to say if the quality is more or less important than the diversity in a constructive method. We are not able to anticipate which can influence more in the application of the local search to the solutions obtained with the construction. We then cannot conclude which of these four methods is the best and will test them with the local search.

3. Improvement Method based on Local Search

We have implemented a two phase local search procedure. The first phase is based on exchanges while the second one is based on insertions.

The neighborhood of the first phase is based on the exchange between a vertex v in the path P and another vertex w not in P (without exceeding the maximum time limit T_{max}). The difference between the scores of both vertices ($S_w - S_v$) provides the associated move value. We examine the vertices in the path and, for each vertex v in P ; we consider to exchange it with each vertex out of the path. We compute the move value of each one and perform the best improving move found. If for a vertex v in P no associated move qualifies to be performed (all of them are non-improving moves),

we try to reduce the length of the path without decreasing the total score. Specifically, we explore again the vertices w not in P but now we focus on those with the same score value than v and check if the exchange reduces the length of the path. In this case we perform the move; otherwise, we resort to the next vertex v in the path P .

When a one-to-one exchange is performed, we try an insertion move in which a vertex not present in the current path is considered to be added to it. Note that in this problem the insertion of a new point into the path could not necessarily increment its length (some points are in the same location). It could even reduce the total length because the matrix distance does not necessarily satisfy the triangular inequality in some instances, and therefore after we add a vertex to the path we have to check the addition of more vertices. This is why after an exchange we consider insertions as long as we can add vertices in the path without exceeding T_{max} . The added vertices are inserted in the best position.

The local search procedure examines the vertices in their order in the path and tries to perform exchanges and insertions for all of them as described above. If a move has been performed for any vertex in the path, we explore again all the vertices in the current path until no further improvement is possible.

3.1 Comparison of Methods

In the previous section we tested sixteen constructive methods (four algorithms with four values of parameter α each one) with 33 instances with optimum known in Fischetti et al. (1998). We identified four of them, C2(0.2), C1(0.6), C1(0.4) and C1(0.2) as the best ones in terms of quality and diversity. Now, we are going to see their performance when the local search is applied to the 100 solutions constructed with each one. For each method and each instance, we compute the best solution found over the 100 constructions plus the local search described above. Table 2 shows the average deviation with respect to the optimum value (Dev) and the number of best solutions that each method is able to match (#best).

	C2(0.2)+LS	C1(0.6)+LS	C1(0.4)+LS	C1(0.2)+LS
Dev	3.96%	5.65%	3.97%	3.42%
#best	16	7	13	17

Table 2. Quality and number of best solutions of constructive methods plus local search.

Table 2 shows that the best solutions are obtained with the C1(0.2)+LS, which is able to achieve a 3.42% deviation and 17 best solutions out of the 33 instances. Anyway, all the methods perform very well considering that their running times are extremely short (below 0.1 seconds on each instance). If we check the best solutions that each method is able to identify, we find out that some of them are obtaining different best solutions. This is especially true when we compare two methods based on different constructive algorithms. For example, if we compare the 16 best solutions obtained with C2(0.2)+LS and the 17 obtained with C1(0.2)+LS, we realize that only 10 of them

are the same and both together are able to match 28 best solutions. As a matter of fact, these are the two methods sharing the least number of best solutions and therefore they are suitable to be combined for improved outcomes.

We have performed a further preliminary experiment to compare the value of the constructive method with the value of the construction coupled with the local search. In particular, we have generated 100 solutions on instance *gil262* with C1(0.2) computed their value, improved them with LS and computed the value of the improved solution. Figure 2 depicts a scatter-plot with 100 points, where the coordinates of each one is the pair of values of each solution (the score of the constructed solution on the x-axis and the score of the improved solution on the y-axis).

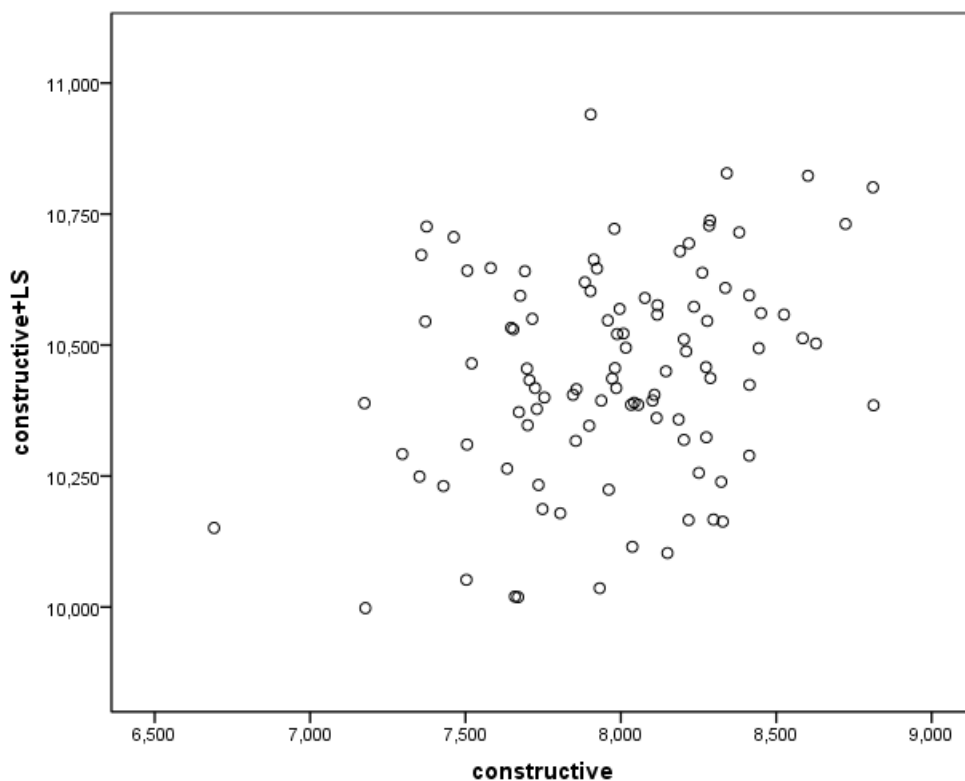


Figure 2. Objective function on *gil262* instance with C1(0.2)

The points in Figure 2 range in the x-axis from 6,691 to 8,813 while in the y-axis they go from 9,998 to 10,940. We observe two effects, the first one is that, as expected, the values in the y-axis are larger than those in the x-axis. The second one is that the range in the y-axis is narrower (942) than the range in the x-axis (2,122). Moreover, the best solution found with the LS with value 10,940 does not come from the best construction value 8,813, which can be interpreted that diversity is as important as quality when apply the local search. However, the correlation coefficient between both variables is $r = 0.284$, which indicates that the correlation is weak, but significantly positive as stated by a *t*-test. Therefore, we cannot conclude with a high confidence that the good solutions of the local search come from the good constructions, and the performance of the entire method is also explained by its diversification power. For the sake of simplicity we only depict here an example

although we have empirically found that this behavior is representative over the instances tested.

4. Path Relinking

Path relinking (PR) was suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover and Laguna, 1997). This approach generates new solutions by exploring trajectories that connect high-quality solutions – by starting from one of these solutions, called the *initiating solution*, and generating a path in the neighborhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions, and incorporating them in an intermediate solution initially originated in the initiating solution.

Laguna and Martí (1999) adapted PR in the context of GRASP as a form of intensification. The relinking in this context consists in finding a path between a solution found with GRASP and a chosen elite solution. Therefore, the relinking concept has a different interpretation within GRASP since the solutions found in one GRASP iteration to the next are not linked by a sequence of moves (as in the case of tabu search). Resende and Ribeiro (2003) present numerous examples of GRASP with PR.

Let $P = \{1, u_1, u_2, \dots, u_k, n\}$ and $Q = \{1, w_1, w_2, \dots, w_s, n\}$ be two solutions (paths from 1 to n) of the orienteering problem. The path relinking procedure $PR(P, Q)$ starts with the first solution P , and gradually transforms it into the second one Q , by swapping out elements in P with elements in Q . The elements in both solutions P and Q , remain in the intermediate solutions generated in the path between them. Let V_{Q-P} be the set of vertices in Q and not present in P and symmetrically, let V_{P-Q} be the set of vertices in P and not present in Q .

To apply $PR(P, Q)$ we order the vertices in V_{Q-P} according to their score, where the vertex with the largest score comes first. In the first step of the path, we insert the first vertex from V_{Q-P} in P in the best position according to the total time of the path. If the obtained path is feasible (its total time does not exceed T_{max}) we have obtained a better solution than P , which is considered the first intermediate solution; otherwise, we remove from this path vertices of V_{P-Q} consecutively until the solution becomes feasible (vertices in V_{P-Q} are selected in increasing score). Let P_1 this feasible intermediate solution. In further steps of the method, we insert into the intermediate solution the next vertex in V_{Q-P} and remove, if necessary, vertices in V_{P-Q} to generate a sequence of feasible solutions until we finally reach Q .

Our implementation of path relinking has two phases. In the first one a set of different solutions is generated with the GRASP method (i.e.: we remove from the set those solutions that are equal). Instead of retaining only the best solution overall when running GRASP, this phase stores all the different solutions obtained with the method. In the second phase we apply the relinking process to each pair of solutions in the set. Given the pair (P, Q) , we consider two paths: from P to Q (where P is the initiating

solution and Q the guiding one), and from Q to P (where they interchange their roles). In short, we apply $PR(P, Q)$ and $PR(Q, P)$.

We have experimentally found that in most cases this relinking process by itself does not produce better solutions than the initiating and guiding solutions. It is convenient to add a local search exploration from some of the visited solutions in order to produce improved outcomes. These results are in line with those reported in Laguna and Martí (1999) for the arc crossing problem. Specifically, we have applied the local search method introduced in the previous section to the best solution generated in the path when it improves either the initiating or the guiding solution, or both.

4.1 Comparing GRASP and GRASP with Path Relinking

In this section we compare GRASP with GRASP with PR on the 33 instances with optimum known in Fischetti et al. (1998) to evaluate the contribution of the path relinking post-processing. Table 3 shows the average value of the objective function, Value, the average deviation with respect to the optimum value (Dev.), the number of optimal solutions (#opt) and best solutions (#best) that each of both methods is able to match, and the CPU time in seconds (Time). We run GRASP for 500 constructions and then apply, in GRASP+PR, the path relinking to all pairs of solutions

	GRASP	GRASP with PR
Value	2078.30	2099.82
Dev.	1.75%	0.84%
#best	14	33
#opt	14	18
Time	1.72	26.87

Table 3. Comparison over 33 instances.

Results in Table 3 clearly indicate that the path relinking post-processing significantly improves the GRASP method. This is especially true with the number of best solutions since GRASP only obtains 14 best values by itself and when it is coupled with PR the combined method is able to match 33 best values. However, Path Relinking consumes much longer running times than GRASP.

To complement the results above, we study now the search profile of our two methods. Figure 3 shows the progression of the best solutions found by our two methods (GRASP and GRASP+PR) for a representative problem instance (cmt200vrpC) during 325.71 seconds of search time. This figure shows how most improvements on the best solution are achieved early in the search (i.e., within 10 percent of the total search time, corresponding to 30 seconds approximately). After that point, GRASP stagnates, while GRASP+PR exhibits an improving trajectory throughout the entire search horizon. In this experiment the best solution found has a score of 2,852 while the optimal value reported in Fischetti et al. (1998) is 2,881 with a CPU time of 17,389.9 seconds on an HP Apollo 9000/720 at 80 MHz, with 58 MIPS, and 18 MFlops. We have performed the same experiment with different instances and the profile

follows the same patter depicted in this figure: GRASP+PR outperforms GRASP over the long search horizon.

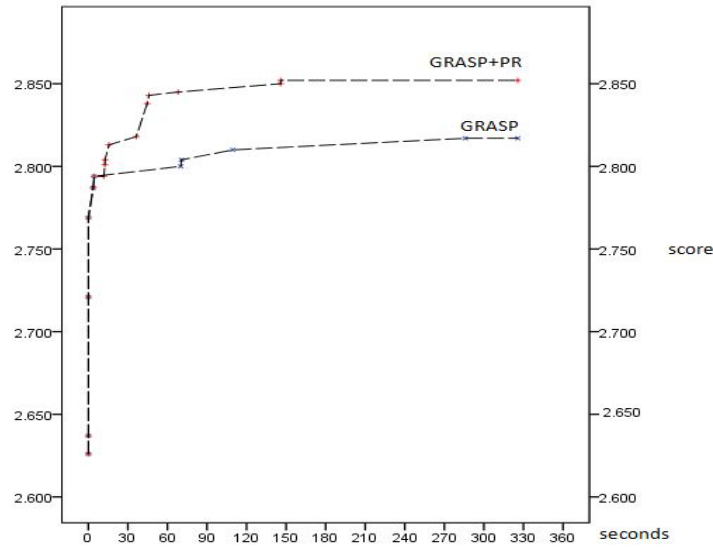


Figure 3. Search profile on instance cmt200vrpC

5. Comparison with Previous Methods

In this section we report our computational experiments to compare the methods proposed in the sections above with the best previous heuristics. Specifically, we compare our GRASP and GRASP with PR with the following four methods:

- CGW by Chao et al. (1996)
- GLS by Vansteenwegen, et al. (2009)
- ACO and VNS by Schilde et al. (2009)

The first benchmark for this problem was proposed by Tsiligirides (1984), which consists of 49 instances with vertices ranging from 21 to 33. These small instances have optimum known and as described in Schilde et al. (2009) most of the current methods are able to match these optima. More recently, Chao et al. (1996) proposed a benchmark with 40 larger instances in two sets: the first set, called p64, has 14 of them with 64 vertices, and the second set, called p66, the other 26 with 66 vertices each one. All the instances in a set are based on the same graph and they only differ in the time limit value (T_{max}). We employ these 40 instances to compare our method with the four previous ones.

Additionally, Fischetti et al. (1998) proposed an exact procedure (a branch-and-cut algorithm) and compute the optimal value for some medium sized instances. As shown in the previous sections, we have used these instances to calibrate our algorithm (find the best values for key search parameters) and compare different search strategies. We also use them in this section to compare the results of our two methods with the optimal solutions.

T_{max}	CGW	GLS	ACO	VNS	GRASP		GRASP+PR	
					value	time	value	time
15	96	96	96	96	96	0.05	96	0.05
20	294	294	294	294	294	0.09	294	0.12
25	390	390	390	390	390	0.14	390	0.31
30	474	474	474	474 / 468	468	0.16	474	0.33
35	570	552	576 / 570	576	576	0.20	576	0.64
40	714	702	714	714	714	0.23	714	0.56
45	816	780	816 / 804	816	810	0.20	816	0.56
50	900	888	900 / 894	900 / 894	900	0.22	900	0.62
55	984	972	984 / 978	984 / 966	978	0.22	984	0.39
60	1044	1062	1062 / 1056	1062 / 1050	1056	0.22	1062	0.42
65	1116	1110	1116	1116	1116	0.22	1116	0.37
70	1176	1188	1188	1188 / 1182	1188	0.22	1188	0.37
75	1224	1236	1236	1236 / 1230	1236	0.22	1236	0.33
80	1272	1260	1284 / 1278	1284 / 1278	1284	0.19	1284	0.23
Avg.	790.7	786.0	795.0 / 792.0	795.0 / 790.7	793.3	0.18	795.0	0.38

Table 4. Comparison of best methods on p64 instances

Tables 4 and 5 show the best values obtained with the four methods referenced above (CGW, GLS, ACO and VNS) on the p64 and p66 sets of instances respectively. We have added to these tables the results of our GRASP and GRASP with Path Relinking (best value found and CPU time in seconds) run for 500 constructions.

Table 4 shows that GRASP obtains a value of 793.3 on 0.18 seconds on average over the 14 p64 instances, while GRASP with Path Relinking obtains an average value of 795.0 on 0.38 seconds. The values of the four previous methods (CGW, GLS, ACO and VNS) are taken from Table 1 in Schilde et al. (2009). CGW and GLS obtain an average value of 790.7 and 786.0 respectively. The ACO and VNS methods are replicated 10 times and Schilde et al. reported the maximum and minimum values across the ten runs (i.e., it seems that they recorded the best solution in each run and then computed the maximum and minimum values of these ten best solutions found). In columns ACO and VNS of Table 3 we report both values when they differ. The averages of these values are 795.0 (max.) and 792.0 (min.) for the ACO and 795.0 (max.) and 790.7 (min.) for the VNS. Therefore, the results obtained with GRASP improve upon those obtained with CGW, GLS and the “min.” runs of ACO and VNS. Moreover, GRASP with Path Relinking is able to match the best results obtained with ACO and VNS. Finally, computing the number of best known solutions that each method is able to match (out of the 14 instances), we obtain: CGW (9), GLS(7), ACO_max(14), ACO_min(8), VNS_max(14), VNS_min(7), GRASP(10), and GRASP with PR(14). Results in Table 4 are in line with these results where the ACO_max, VNS_max, GRASP and GRASP with PR are able to match the 26 best known results for these 26 instances.

Note that we are only reporting running times of our two methods in Tables 4 and 5, obtained on our Intel Pentium I5 at 3.20 GHz. As mentioned, the values of the four previous methods in these tables are taken from Tables 1 and 2 in Schilde et al. (2009) respectively, and correspond to an Intel Pentium 4D at 3.2 Ghz and 4 GB of RAM. In those tables running times are missing for CGW and GLS methods, and they report

1.97 elapsed seconds on average for ACO and 2.57 for VNS to reach the best solution found. Note that these are not the total running times of the methods as we are reporting for GRASP and GRASP+PR, but only a fraction of them. Moreover, these methods are replicated ten times and apparently the running times reported correspond to a fraction of a single execution (the one in which the best solution was found). Therefore, considering that our GRASP with Path Relinking consumes 0.38 seconds on average (the entire execution), it seems that the two best previous methods need much longer running times to obtain the same high quality solutions.

T_{max}	CGW	GLS	ACO	VNS	GRASP		GRASP+PR	
					(value)	(time)	(value)	(time)
5	10	10	10	10	10	0.031	10	0.031
10	40	40	40	40	40	0.031	40	0.031
15	120	120	120	120	120	0.046	120	0.046
20	195	175	205	205	205	0.046	205	0.078
25	290	290	290	290	290	0.078	290	0.187
30	400	400	400	400	400	0.124	400	0.374
35	460	465	465	465	465	0.140	465	0.686
40	575	575	575	575	575	0.140	575	0.592
45	650	640	650	650	650	0.156	650	1.092
50	730	710	730	730	730	0.187	730	1.123
55	825	825	825	825	825	0.218	825	1.216
60	915	905	915	915	915	0.218	915	0.920
65	980	935	980	980	980	0.218	980	1.482
70	1070	1070	1070	1070	1070	0.234	1070	1.372
75	1140	1140	1140	1140 / 1135	1140	0.249	1140	1.684
80	1215	1195	1215	1215 / 1195	1215	0.265	1215	1.092
85	1270	1265	1270	1270 / 1265	1270	0.249	1270	1.513
90	1340	1300	1340	1340 / 1330	1340	0.249	1340	1.248
95	1380	1385	1395	1395 / 1390	1395	0.249	1395	1.279
100	1435	1445	1465	1465 / 1445	1465	0.249	1465	0.951
105	1510	1505	1520	1520 / 1495	1520	0.234	1520	1.029
110	1550	1560	1560	1560 / 1550	1560	0.218	1560	0.795
115	1595	1580	1595	1595 / 1585	1595	0.218	1595	0.748
120	1635	1635	1635	1635 / 1625	1635	0.187	1635	0.546
125	1665	1665	1670	1670 / 1665	1670	0.171	1670	0.421
130	1680	1680	1680	1680	1680	0.031	1680	0.031
	948,7	942,9	952,3	952.3/947.5	952.3	0.171	952.3	0.791

Table 5. Comparison of best methods on p66 instances

Results in Table 5 are in line with those reported in Table 4 although we observe that now our two methods, GRASP and GRASP with PR, obtain the same solutions. This may indicate that these instances, p66, are easier to solve than those reported in Table 4, p64. Moreover, the ten runs of ACO provide the same solution, which seems to confirm it. On the other hand, ACO, GRASP and GRASP with PR outperform CGW and GLS on these instances.

We now consider 33 (OP and VRP) instances reported in Fischetti et al. (1998) for which the optimum solution is known. Note that in the VRP instances the customer demand is interpreted as the vertex score. Table 6 reports, for each instance, the optimal value (opt) and the value of GRASP and GRASP with Path Relinking with the associated running time in seconds. Since some of these instances are larger than those in the previous experiment, we consider here two runs of our two methods, one with 500 constructions (labeled as GRASP and GRASP+PR respectively) and another with 1000 constructions (labeled as GRASP10 and GRASP10+PR respectively).

instance	opt	GRASP		GRASP+PR		GRASP10		GRASP10+PR	
		value	time	value	time	value	time	value	time
op33A	250	250	0.02	250	0.17	250	0.06	250	0.30
att48vrpA	13	13	0.05	13	0.09	13	0.12	13	0.19
eil30vrpA	2650	2650	0.00	2650	0.02	2650	0.03	2650	0.03
eil51vrpA	264	264	0.05	264	0.37	264	0.12	264	0.84
eil76vrpA	490	490	0.19	490	2.96	490	0.06	490	7.30
eil101vrpA	572	572	0.41	572	9.83	572	0.12	572	23.85
cmt101vrpA	530	520	0.19	530	2.42	520	0.03	530	5.01
cmt121vrpA	412	408	0.70	412	22.29	408	0.12	412	50.19
cmt151vrpA	824	811	1.26	824	19.91	814	0.39	824	58.05
cmt200vrpA	1205	1162	3.04	1181	36.13	1162	0.80	1182	113.49
gil262vrpA	4466	4105	5.68	4342	77.36	4167	0.36	4342	284.88
op33B	500	500	0.05	500	0.97	500	1.40	500	2.59
att48vrpB	30	30	0.19	30	2.65	30	2.51	30	6.18
eil30vrpB	7600	7600	0.02	7600	0.05	7600	6.07	7600	0.09
eil51vrpB	508	508	0.12	508	1.98	508	11.31	508	4.60
eil76vrpB	907	892	0.36	904	8.92	897	0.11	907	24.04
eil101vrpB	1049	1020	0.69	1032	12.25	1020	0.34	1032	34.59
cmt101vrpB	1030	990	0.44	1020	11.33	990	0.05	1020	28.75
cmt121vrpB	715	702	0.86	707	15.88	702	0.22	707	44.12
cmt151vrpB	1537	1481	2.25	1519	26.79	1501	0.70	1526	86.31
cmt200vrpB	2198	2059	5.60	2102	45.26	2076	1.39	2103	147.75
gil262vrpB	8456	7896	11.26	8004	95.22	7896	0.86	8004	341.86
op33C	660	660	0.06	660	1.45	660	1.73	660	4.32
att48vrpC	39	39	0.19	39	3.57	39	4.52	39	9.19
eil30vrpC	11550	11550	0.03	11550	0.14	11550	11.17	11550	0.34
eil51vrpC	690	690	0.12	690	2.39	690	22.74	690	5.13
eil76vrpC	1186	1180	0.37	1181	6.27	1180	0.09	1181	15.21
eil101vrpC	1336	1295	0.66	1300	11.81	1295	0.36	1304	31.70
cmt101vrpC	1480	1450	0.58	1480	16.07	1450	0.09	1480	41.06
cmt121vrpC	1134	1108	1.34	1119	45.36	1108	0.25	1119	116.84
cmt151vrpC	2003	1973	2.36	1992	45.54	1973	0.73	1992	137.37
cmt200vrpC	2881	2794	5.74	2850	94.67	2794	1.33	2852	318.99
gil262vrpC	11195	10922	11.90	10979	266.56	10922	1.15	11019	1043.63
Average	2132.12	2078.30	1.72	2099.82	26.87	2081.55	3.43	2101.58	90.57

Table 6. Comparison with optimal values

GRASP is the quickest method with an average running time of 1.72 seconds and an average deviation from optimality of 1.75%, on the other hand GRASP10 takes approximately twice the average time of GRASP, reducing only 0.14% the average deviation with respect to optimal values. Comparing GRASP+PR with GRASP10+PR, we can observe that the second method employs 90.57 seconds on average and exhibits an average deviation from optimality of 0.79% whilst the first one consumes on average only 26.87 seconds with an average deviation of 0.84%. At this point, it seems that a further increasing in the number of constructions would not contribute substantially to reduce the deviation from optimality.

6. Conclusions

The orienteering is a difficult combinatorial optimization problem and a perfect platform to study the effectiveness of search mechanisms. Of particular interest in our work has been testing the effect of combining two different neighborhoods within the local search of GRASP as well as studying the effect of a post-processing based on Path Relinking. Through extensive experimentation, we have been able to determine the benefits of adding enhanced search strategies to basic procedures. We have compared our methods with the best identified in previous studies. The comparison favors our proposal.

Acknowledgments

This research has been partially supported by the *Ministerio de Educación y Ciencia* of Spain (Grant Ref. TIN2009-07516).

References

- Chao, I.M., Golden, B.L. and Wasil, E.A. (1996) A fast and effective heuristic for the orienteering problem, *European Journal of Operational Research* 88, 475-489.
- Fischetti, M., Salazar, J.J. and Toth, P. (1998) Solving the orienteering problem through branch and cut, *INFORMS Journal on Computing* 10, 133- 148.
- Glover, F. , Laguna, M. (1997) Tabu Search, Kluwer Academic Publishers, Boston.
- Laguna, M. and Martí, R. (1999) GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52.
- Miller, C. E., A. W. Tucker, and Zemlin, R. A. (1960) Integer programming formulations and traveling salesman problems, *Journal ACM*, 7, 326–329.
- Resende, M.G.C. and Werneck, R.F. (2004) A hybrid heuristic for the p-median problem, *Journal of Heuristics*, 10:59–88.

- Resende, M.G.C. and Ribeiro, C.C. (2003) Greedy randomized adaptive search procedures. F. Glover, G. Kochenberger, eds. *State-of-the-art Handbook in Metaheuristics*, Kluwer Academic Publishers, Boston, MA. 219–250.
- Schilde, M., Doerner, K.F., Hartl, R.F. and Kiechle, G. (2009), Metaheuristics for the bi-objective orienteering problem, *Swarm Intell.*, 3, 179 -201.
- Souffriau, W., Vansteenwegen, P., Vanden Berghe, G. and Oudheusden, D.V. (2010) A path relinking approach for the team orienteering problem, *Computers and Operations Research*, 37, 1853-1859.
- Tsiligirides, T. (1984) Heuristic methods applied to orienteering, *Journal of the Operational Research Society*, 35(9), 797—809.
- Vansteenwegen, P., Souffriau, W. and Oudheusden, D.V. (2011) The orienteering problem: A survey, *European Journal of Operational Research* 209, 1-10.
- Vansteenwegen, P., Souffriau, W., Vanden Berghe, G. and Oudheusden, D.V. (2009) A guided local search metaheuristic for the team orienteering problem. *European Journal of the Operational Research*, 196(1), 118-127.