An Hybrid GRASP+VNS Metaheuristic for the Prize-Collecting Traveling Salesman Problem

Leonardo M. Gomes, Viviane B. Diniz, Carlos A. Martinhon (*)[e-mail]

Departamento de Ciência da Computação / Instituto de Computação
Universidade Federal Fluminense (UFF) - Niterói, RJ, Brasil

Abstract: In the Prize-Collecting Traveling Salesman Problem (PCTSP) we have to determine a tour visiting each vertex in the graph at most one time. If a given vertex is selected then an associated prize is collected, if a vertex is unrouted a penalty must be paid. We want to minimize an objective function balancing between the travel cost and the total penalties in a such way that a sufficiently large prize is collected. In this paper we present an hybrid metaheuristic that combines Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neigboorhood Search procedure as a local search. Experimental results show that it is potentially a powerful heuristic device, since it greatly enhace different features of these two approaches.

Key words: Greedy Randomized Adaptive Search Procedure - GRASP, Variable Neighborhood Search - VNS, Local Search.

1 Introduction

In the Prize-Collecting Traveling Salesman Problem (PCTSP) a prize is acquired in every visited city and a penalty is paid for every unrouted city. We want to mimimize the sum of travel costs and net penalties, while including in this tour enough cities to collect a prescribed amount of money (or prize). The generated tour must visit each city at most one time.

Since the PCTSP generalizes the well known Traveling Salesman Problem (TSP) it is also an NP-hard problem (see Garey&Johnson [1979]). Observe that the PCTSP coincide with TSP if all node associated penalties are infinite.

The PCTSP was introduced by Balas and Martin[1985] as a model for scheduling the daily operation of a steel rolling mill. A rolling mill, produces steel sheet from slabs by hot or cold rolling. For reasons that have to do with the wear and tear of the rolls as well as other factors, the sequence in which various orders are processed is essential. Scheduling a round consists of choosing from an inventory of slabs assigned to orders, a collection that satisfies a lower bound on total weight, and ordering it into an appropriate sequence, e.g., one that minimizes some function of the sequence. Since the choice of slabs for the round limits the options available for their sequencing, the two tasks must be solved jointly. The PCTSP captures the essential features of this problem.

Balas[1989,1993] have been presented some structural properties of the PCTSP polytope. Families of facet inducing inequalities are identified, some of which are related to the TSP polytope and others with the knapsach polytope. Fischetti and Toth[1988] developed bounding procedures

(*) e-mail: mart@dcc.ic.uff.br

based on different relaxations. Lower bounds for the asymmetric version of the PCTSP are presented by Dell'Amico, Maffioli and Värbrand[1994].

Goemans-Willianson[1992] provides a 2-approximation procedure to a version of the PCTSP in which the minimum prize to be collected is removed. That is, the goal is simply to minimize the cost of the tour as discussed above. To approximate the PCTSP of Balas; Awerbuch, Azar, Blum and Vempala[1995] developes the first approximation algorithm for the PCTSP having a poly-logarithmic performance.

Before describing our basic procedure and variants, we provide background on both GRASP and VNS in the following two sections. Section 4 presents a detailed description of our combined GRASP+VNS metaheuristic. Section 5 presents the empirical results and concluding remarks are given in section 6.

2 Greedy Randomized Adaptive Search Procedure (GRASP)

Consider a finite but large set S of arbitrary objects. Combinatorial optimization problem consist in finding $x^* \in X \subseteq S$ such that some objetive function f is minimized, i.e. $f(x^*) = \min\{f(x): x \in X, X \subseteq S\}$. The symbols S, X, x and f denotes respectively the solution space, feasible set, feasible solution, and real valued function.

The Greedy Randomized Adaptive Search Procedures, proposed by Feo and Resende[1995] are basically composed by two different phases: a construction phase, in which a feasible solution is produced and a local search phase, in which a local minimum is obtained using the feasible solution generated in the first procedure. The best overall solution is kept as the final result.

To build up one starting solution in the *construction phase*, one element at a time is selected. The elements are initially ordered in a candidate list with respect to a greedy function previously defined. This function measures the (myopic) benefit of selecting each element. This procedure is adaptive since the benefits associated with every element are updated at each iteration of the construction phase reflecting the changes brought on by the selection of the previous element. The best elements in the candidate list are considered to build up a new *Restricted Candidate List* - RCL. The probabilistic component of a GRASP is characterized by a random choice of the element that is not necessarily the top candidate of the RCL. This choice technique allows for different solutions to be obtained at each GRASP iteration, but does not necessarily compromise the strength of the adaptive greedy component of the method.

There is no guarantee that the solution generated in the construction phase is locally optimal. Hence, a *local search* procedure is applied generating a sequence of neighbours that improve the initial solution. Therefore a neighborhood structure that associates a feasible solution x to a subset of solutions $N(x)$ must be constructed. The efficiency of the local search procedure basically depends of the suitable choice of a neighborhood structure and the starting point generated at the construction phase.

It is difficult to formally analyze the quality of different solution values achieved by sucessive iterations of the GRASP procedure. Nevertheless, based on empirical observations, it has been found that the resulted sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction, but the over all trials dominates the deterministic solution with a high probability (see Feo and Resende [1995]).

As sumarized by Resende[1998], a number of insteresting enhancements to the basic GRASP can be applied. Different strategies which deals with path relinking, reactive GRASP, long-term memory, the proximate optimality principle, paralell GRASP can be considered. A more detailed description and bibliographies related to these improvements can be found at Resende[1998].

3 Variable Neighborhood Search (VNS)

Variable Neighborhood Search - VNS (see Mladenovic and Hansen [1997]), systematicly change neighborhoods during the local search procedure. Contrary to other metaheuristics based on local search methods, VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solution and jumps from this solution to a new one if and only if an improvent is attained.

To rapidly expose the main steps of VNS, consider a finite set of pre-selected neighborhood structures with N_k, (k=1,..,k_{max}), and with $N_k(x)$ the set of solutions in the k^{th} neighborhood of *x*. Given an initial point $x \in X$ the basic VNS heuristic comprises the following steps:

Procedure VNS: Given N_k, (k=1,..,k_{max}) and some $x \in X$

```
01. Begin
02.     Repeat
03.      k = 1;
04.         Repeat
05.             Generate at random x'∈ N_k(x);
06.             x"← apply local search with x' as starting point;
07.             if ( f(x") < f(x) ) then
08.                 x ←x";
09.                 k ←1;
10.             else
11.                 k←k+1;
12.         until k = k_max
13.     until (some stop condition is satisfied);
14. end.
```

Figure 3.1: Variable Neighborhood Search

The stopping condition used, may be e.g. the maximum number of iterations, maximum CPU time allowed or maximum number of iterations between two improvements.

Interesting enhancements may be considered in addition to the basic VNS procedure. Hansen and Mladenovic[1998], suggests the use of descent-ascent steps allowing worst neighbours with some probability. Hansen et al. [1998] propose the Variable Neighborhood Decomposition Search - VNDS for dealing with large instances of various problems.

As appointed by Mladenovic and Hansen[1997], often successive neighborhoods N_k will be nested. Nevertheless, also distinguished neighborhoods may be considered exploring different features of the feasible set (not necessarilly in a nested fashion).

As a local optimum within some neighborhood is not necessarily one within another, a change of neighborhoods can be performed during the local search phase too. This local searh is then called Variable Neighborhood Descent (VND). Its steps are described in the figure 3.2. Observe that since no improvement is attained a new neighborhood is selected until the last neighborhood $N_{k_{max}}(x)$ is reached.

In order to apply a based VNS or VND as a local search (in our GRASP procedure) we carefully selected neighborhoods exploring different features of the feasible set. The next section give us a detailed description of our hybrid procedure applied to the PCTSP

Procedure VND: Given N_k, (k=1,.., k_{max}) and some x∈ X
01. **Begin**
02. **Repeat**
03. k=1;
04. **Repeat**
05. x' ← local-search(x , $N_k(x)$);
06. **if** (f(x') < f(x)) **then** x ←x'
07. **else** k←k+1;
08. **until** $k = k_{max}$;
09. **until** (some stop condition is satisfied)
10. **end.**

Figure 3.2: Variable Neighborhood Descent

4 An Hybrid GRASP+VNS for the Prize-Collecting TSP

Prior to the presentation of our hybrid procedure some basic definitions are necessary. Let $G = (V_0, E)$ denote a complete non-directed graph with a vertex set V_0 with n+1 vertices indexed {0,1,2,..,n} and an edge set E. Futhermore assume $V \subset V_0$ to be the subset of vertices indexed by {1,2,..,n}. All vertices in V are associated with customers and the vertex indexed by 0 is associated with the depot or a starting point. To each edge (i,j) beloning to E one has travel costs c_{ij} which may express time, distance or a combination of both. Let M be the minimum amount of money (prize) to be collected, w_i be the associated prize to be paid if vertex $i \in V_0$ is visited and γ_i the corresponding penalty if vertex $i \in V_0$ is unrouted. We take $\gamma_0 = \infty$ to force visitation of node 0. We do not take into account the prize assotiated to the depot (i.e $w_0 = 0$). The sequence of visited nodes will be represented by α and E(α)⊆E denotes the set of edges with both extremities in α. Travel costs and the collected prize assotiated to route α will be given respectively by T_α and P_α.

Given α, let k be an arbitrary vertex not beloning to α. We will denote by $g_\alpha(k)$ the resulted gain at the objective function value after the addition of node k to route α. Then:

$$g_\alpha(k) = \max_{(i,j)\in E(\alpha)} \{c_{ij} + \gamma_k - c_{ik} - c_{kj}\} \qquad (4.1)$$

where c_{ij}, c_{ik} and c_{kj} are the corresponding travel costs and γ_k the penalty to be paid if node k is unrouted.

Observe that, if α is a feasible route the assotiated prize P_α is greater than M (the minimum amount of prize to be collected). These gains are computed and stored in a vector L. It is easy to observe that the objective function value can be decreased if $g_\alpha(k)$>0 is considered for some node k∉ α. Using L a Restricted Candidate List RCL_λ is obtained only using the best λ values of L where $0 \le \lambda \le 1$. The RCL_λ is computed considering values in the interval $[(1-\lambda)(g_{max} - g_{min}) + g_{min}, g_{max}]$ where g_{max} and g_{min} denotes respectively the maximum and minumum values of L. Observe that λ=0

implies in a purely greedy choice since RCL_λ only contains the g_{max} element. On the other side, if λ=1, RCL_λ and L are identical.

Our algorithm is summarized in the figure 4.1. The parameters *GIter* denotes the overall GRASP iterations while *CIter* represents the total number of initial starting points (at the construction phase) before the local search procedure.

Procedure: (GRASP+VNS/ Prize-Collecting TSP)
01 . Begin
02 . $T_\beta \leftarrow \infty$; {value of the best encountered solution}
03 . $\bar{\rho} \leftarrow 0$; $P_{\bar{\rho}} \leftarrow 0$; {find an initial solution $\bar{\rho}$}
04 . Compute: $T_{\bar{\rho}} = \sum_{i \in \bar{\rho}} \gamma_i$; {initialize value of the objective function}
05. **For** (each k$\notin \bar{\rho}$) **do**
06. $g_{\bar{\rho}}(k) = \max_{(i,j) \in E(\bar{\rho})} \{c_{ij} + \gamma_k - c_{ik} - c_{kj}\}$;
07. Compute: $\bar{L} = \{ g_{\bar{\rho}}(k) \ \ for\ each\ k \notin \bar{\rho} \}$;
08. Obtain the $\overline{RCL}_\lambda$ using the best λ values of $\bar{L}$;
09. **For** (i=1,...,GIter) **do**
10. $T_\alpha \leftarrow \infty$; {initialize value of the best solution in the construction phase}
11. **For** (j=1,...,CIter) **do** {generate a set of starting points - filter}
12. $T_\rho \leftarrow T_{\bar{\rho}}$; $\rho \leftarrow \bar{\rho}$; $P_\rho \leftarrow P_{\bar{\rho}}$;
13. $L \leftarrow \bar{L}$; $RCL_\lambda \leftarrow \overline{RCL}_\lambda$;
14. **Repeat**
15. Select at random a value of RCL_λ. Let $\bar{k} \in$ V the associated node;
16. Insert $\bar{k}$ in route ρ; {update ρ}
18. $P_\rho \leftarrow P_\rho + w_{\bar{k}}$;
19. $T_\rho \leftarrow T_\rho - g_\rho(\bar{k})$;
20. Update L;
21. Update RCL_λ;
22. **Until** $g_\rho(\bar{k}) \le 0$ *and* $P_\rho \ge M$;
23. **If** ($T_\rho < T_\alpha$) **then** {save the best starting point in the construction phase}
24. $T_\alpha \leftarrow T_\rho$; $\alpha \leftarrow \rho$; $P_\alpha \leftarrow P_\rho$;
25. **end if**;
26. **end for**;
28. Local-search $(\alpha, T_\alpha, P_\alpha)$; {using VNS (or VND) - return a new α *and* T_α}
29. **If** ($T_\alpha < T_\beta$) **then** {update the best overall solution}
30. $\beta \leftarrow \alpha$, $P_\beta \leftarrow P_\alpha$ and $T_\beta \leftarrow T_\alpha$;
31. **end for**;
32. **End.**

Figure 4.1: GRASP+VNS/Prize-Collecting TSP

At the filtering algorithm (comprising lines 11-26), the construction phase is executed by *CIter* iterations and the best encountered solution is gathered as initial point to our based VNS or VND local search. This idea was successful used by Laguna and Martí[1997] for coloring sparse graphs. Note that, CIter = 1 corresponds to the standard GRASP construction phase.

The combined GRASP+VNS procedure, starts considering node 0 as initial route and then perform a sequence of ADD-steps until feasibility is reached and no more positive gains $g_\rho(k)$ (for $k \notin \alpha$) are found. Obviously, the opposite can also be done considering an initial TSP solution and then executing a sequence of DROP-steps always preserving feasibility.

To apply the local search using the VNS methodology, a set of distinct neighborhood structures must be considered. In our approach, the main idea is to explore different features of the solution space. The implemented based VNS local seach procedure is composed by three different neighborhood structures.

The classical 3-optimal (or 2 optimal) procedure is used to define the first neighborhood (S. Lin [1965], Christofides and Eilon [1972]). Observe that this edge-exchange procedure mantains the same set of nodes gathered in the construction phase. Clearly, 3-optimal or 2-optimal procedures will terminate at a local optimum (not necessarily a global optimum), thus producing an approximate solution.

In order to define the other two neighborhood structures, two types of movements are first considered: DROP-step and ADD-step respectively. Using the DROP procedure we can apply a sequence of node removals (from route α) while the objective function value is being decreased. Analogously, using the ADD procedure, a sequence of node additions can be performed while some improvement is attained. These two procedures will be alternately used in a convenient way to define neighborhoods 2 and 3.

To describe the DROP procedure, let us define respectively $p_k, s_k \in \alpha$ be the predecessor and sucessor of node $k \in \alpha$.

Procedure: DROP-steps $(\alpha, T_\alpha, P_\alpha)$;

01. Begin

02. **For** (each $k \in \alpha$) **do** {compute the reduction in the obj. function for each k}

03. $h_\alpha(k) = c_{p_k,k} + c_{k,s_k} - c_{p_k,s_k} - \gamma_k$;

04 Let $L = \{h_\alpha(k) / w_k \ \textit{such that}\ P_\alpha - w_k \geq M \ \textit{and} \ k \in \alpha\}$;

05. **Repeat**

06. Select: $\bar{k} = \arg max\{h_\alpha(k) / w_k \ \textit{for each}\ k \in L\}$;

07. **If** $h_k(\bar{k}) > 0$ **then**

08. Remove $\bar{k}$ to route α; {update α}

09. $P_\alpha \leftarrow P_\alpha - w_{\bar{k}}$ and $T_\alpha \leftarrow T_\alpha - h_\alpha(\bar{k})$;

10. Update L;

11. **end if**;

12. **Until** $L = \varnothing$;

13. End.

Figure 4.2: DROP-steps with best improving/ Prize Collecting TSP

Observe that, if one applies a DROP movement to some feasible route α the updated prize must remain greater or equal to *M* (the miminum amount of prize to be collected). Analogously to the greedy algorithm for the continuous 0-1 knapsak problem (see for example, Horowitz and Sahni

[1978]), a balance between $h_\alpha(k)$ and w_k (for k$\in\alpha$) is taken into account. A large improvement in the objective function value simply considering $h_\alpha(k)$ may be disadvantageous if the corresponding prize w_k is too large. The algorithm in the figure 4.2 ilustrates the use of a *best-improving* strategy, since we look at all neighbours of a given solution and then move to the best (steepest descent). The *first-improving* strategy can also be considered. In this case, we look at the neighbours one by one, taking them in a random order, and moving to the first one that improves the current solution. Details of these two strategies are discussed by Anderson[1996].

Note that, only nodes whose removal does not imply infeasibility are considered. As we shall see later, to define distinct neighborhoods, DROP-steps with negative gains or even defining infeasible solutions can also be performed.

Given a route α, T_α (cost) and P_α (assotiated prize), the ADD procedure can be described as follows :

Procedure: ADD-steps($\alpha, T_\alpha, P_\alpha$);
01. Begin
02. **For** (each k$\notin\alpha$) **do** {compute the increasing in the obj. function value for each k}
03. $g_\alpha(k) = \max_{(i,j)\in E(\alpha)} \{c_{ij} + \gamma_k - c_{ik} - c_{kj}\}$;
04. Let $L = \{g_\alpha(k)$: *for each* $k \notin \alpha\}$;
05. **Repeat**
06. Select: $\bar{k} = \arg max\{g_\alpha(k)$ *for each* $k \notin \alpha\}$
07. **If** $g_k(\bar{k}) > 0$ **then**
08. Insert $\bar{k}$ to route α; {update α}
09. $P_\alpha \leftarrow P_\alpha + w_{\bar{k}}$ and $T_\alpha \leftarrow T_\alpha - g_\alpha(\bar{k})$;
10. Update L;
11. **end if**;
12. **Until** $L = \varnothing$;
13. End.

Figure 4.3: ADD-steps with best -improving/ Prize Collecting TSP

Like DROP, the ADD procedure also defines a greedy heuristic. Observe that, if $g_\alpha(k) > 0$ for some k$\notin\alpha$, a decreasing in the objetive function value will be done. Analogously to the DROP-step procedure the first-improving strategy can also be considered. Figure 4.3 ilustrates the use of the best-improving strategy.

To define the second neighborhood structure (node-exchange1) we alternately consider a sequence of DROP and ADD movements. The main aspect to be observed is that all movements are executed preserving feasibility. A sequence of feasible DROP movements is considered until no improvement in the objetive function value is attained. After that, a sequence of ADD movements is performed until is possible (using the first or best improving strategy). Each iteration in the node-exchange1 correspond to a sequence of feasible DROP-steps followed by a sequence of feasible ADD-steps. Another examples, in which DROP and ADD-steps are used to define new neighborhoods are given by Voβ[1996] for the Traveling Purchaser Problem.

In the third neighborhood (node-exchange2) only DROP movements with $h_\alpha(k) < 0$ are considered. Some carefull however, must be taken. Assotiated to an infeasible removal, a sequence of the best additions (not using node k) must be done restoring feasibility. This idea remind the

strategic oscillation present in the tabu search algorithm (see Glover [1989], [1990]). If one DROP movement does not destroy feasibility, a sequence of positive ADD's (with $g_k(k) > 0$, for k$\notin \alpha$) is done until is possible. In this case, each iteration corresponds to a single DROP-step (feasible or not) followed by a sequence of node aditions restoring feasibility when necessary.

The local search using our based VNS procedure is summarized in the figure 4.4:

Procedure: Local seach -VNS($\alpha, T_\alpha, P_\alpha$)

01. **Begin**
02. **do**
03. **do**
04. $\alpha \leftarrow 3 - optimal(\alpha, T_\alpha)$; {*First Neighborhood* / Edge-exchange}
05. **Repeat** {*Second Neighborhood* / Node-exchange1}
06. execute DROP-steps as described in the figure 4.2
07. **If** some removal was performed **then**
08. apply ADD-steps until is possible;
09. **until** no ADD movements are performed;
10. **while** some improvement in the second neighborhood is attained;
11. $H = \{k \in \alpha / h_\alpha(k) < 0\}$; {*Third Neighborhood* / Node-exchange2}
12. **while** H $\neq \varnothing$ **do**
13. select an arbitrary $\bar{k} \in H$;
14. $H \leftarrow H - \{\bar{k}\}$;
15. $\xi \leftarrow \alpha$; $T_\xi \leftarrow T_\alpha$ and $P_\xi \leftarrow P_\alpha$;
16. remove $\bar{k}$ from ξ; {DROP movement}
17. **If** $P_\xi - w_{\bar{k}} < M$ **then**
18. apply ADD-steps until feasibility is restored;
19. perform feasible ADD-steps while $g_\xi(l) > 0$ for $l \notin \xi$;
20. **If** $T_\xi < T_\alpha$ **then**
21. $\alpha \leftarrow \xi$; $T_\alpha \leftarrow T_\xi$ and $P_\alpha \leftarrow P_\xi$; {update α}
22. update H;
23. **end if;**
24. **end while**;
25. **while** some improvement in the third neighborhood is attained**;**
26. **end.**

Figure 4.4: Based VNS Local Searh / Prize Collecting TSP

Only feasible moves decreasing the objetive function values are used to define the second neighborhood (node-exchange1). As a consequence, all changes are performed directly on route α. The same does not happen with node-exchange2. Since a single DROP-step (with $h_\xi(k) < 0$ for some $k \in \xi$) is applied, the undesirable increasing in the objective function value is repaired with a sequence of node additions. If the resulting cost T_ξ is less than T_α; we updadte α, P_α and T_α respectively. If $T_\xi \geq T_\alpha$, route ξ is rejected. One shoud notice that, since nodes to be removed are

chosen at random, the described procedure (at the third neighborhood) corresponds to the first improving strategy.

Observe that, each time we perform a single iteration in the second or third neighborhood (node-exchange procedures), different number of ADD-DROP steps are necessary. A large number of node additions or removals at the current route will probably result in a non organized sequence of nodes. The returning to the first neighborhood (edge exchange procedure) is justified specially if the total number of single ADD-DROP movements is large. To reduce the number of times the edge-exchange is performed, a parameter θ may be introduced in order to allow 3-optimal (or 2-optimal) procedure if and only if the total number of ADD and DROP steps is larger than θ.

A based VND local search procedure has also been considered. In addition to the three neighborhoods used, another edge-exchange procedure (2-optimal) was implemented. The sequence of selected neighborhoods are as follows: 2-optimal, node-exchanges 1 and 2 (as discussed above) and finally 3-optimal procedure. Some computational results "comparing" the use of the based VNS and VND as local search as well as the the use of filtering procedure at the GRASP construction phase are listed at section 5.

5 Computational Results

Computational experiments have been conducted in order to evaluate the quality of the upper bounds proposed in this paper. These experiments were performed on a Pentium III 450Mz with 128 of RAM Memory under the Linux operating system . The algorithms were coded in C and the option *gcc <arc>.c -o <arc> -lm -O3* was activated for compilation.

Text problems for the Prize-Collecting TSP were not encountered at the avaliable literature. Consequently a set of instances were generated at random (see table 5.1). The coordinates of each point have been taken from the TSPLIB (*http:/www.iwr.uni-heidelberg.de/iwr/comopt/ software/ TSPLIB 95*). In all cases we simply added new columns at random defining assotiated prizes and penalties.

Table of figure 5.1 ilustrates the use of two different versions of the GRASP construction phase. The first one applies a construction phase without filtering while the second one filters the best solution to be used in the local search. In both GRASP+VNS algorithms, we set *GIter=500* and λ=0.20 respectively. Computational time expended in both cases (with and without filter) as well as the associated objective function values are listed.

(1)		(2)	(3)	(4)	(5)	(6)
Problems		**Min Prize**	**GRASP+VNS *without* filtering**	**cpu time (secs)**	**GRASP+VNS *with* filtering**	**cpu time (secs)**
pc50a		400	380,45	19,85	380,45	**19,84**
pc50b		500	380,45	21,03	380,45	**20,74**
pc50c		600	381,38	27,67	381,38	**25,82**
pc50d		700	**393,59**	47,92	394,57	**40,51**
pc69a		1000	672,33	209,90	672,33	**167,89**
pc75a		900	503,61	245,76	503,61	**212,57**
pc75b		1050	**503,34**	251,76	503,61	**215,99**
pc100		550	**644,79**	1213,50	645,15	**1025,49**
pc120a		1000	654,03	1577,44	654,03	**1333,33**
pc130		1000	654,73	2388,20	**653,87**	**2059,68**
pc136		2500	685,08	3418,41	685,08	**2940,85**
pc266		5000	1205,44	47972,29	**1195,13**	**41.387,35**
pc360a		7000	**2150,47**	78499,19	2159,32	**78.460,55**
pc360b		9000	10.383,69	121989,31	**10190,41**	**113543,83**

Figure 5.1: Filtering procedure at the GRASP construction phase

The 3-optimal algorithm was applied to define the first neighborhood (edge-exchange procedure). Column (1) exibits the name of each instance as well as the total number of customers to be considered. Column (2) shows the miminum amount of prize to be collected. Column (3) shows the performance of the GRASP+VNS without filtering (CIter = 1) and column (4) exibits the corresponding computational effort. Finally, columns (5) and (6) ilustrates the use of the filtering procedure (CIter = 10) and the associated computational performance.

Note that, despite the computational effort expended at the GRASP construction phase using filtering procedure, a lower computational time is observed compared to those ones listed at column (3) (without filtering). The computational effort evolved at the based VNS local search, may be mininimized if a good starting point is used. One should notice that, bad starting points allows for a great number of local search iterations.

Table of figure 5.2 ilustrates the use of two different versions of the based VNS procedure (using 2-opt and 3-opt respectively) and a version of the based VND local search. All cases were performed using the filtering procedure at the GRASP construction phase. The parameter CIter = 10 was applied in all situations. Columns (1) and (2) are as defined before (figure 5.1). Column (3) shows the objective function values when the 3-optimal procedure (used at the first neighborhood) is changed by the 2-optimal procedure. The objective function values (using 3-opt) are listed at column (5). Column (7) exibits the performance of the hybrid GRASP+VND algorithm. As discussed before, the selected neighborhoods at the based VND local search are: 2-optimal, nodes-exchange 1 and 2 and finally the 3-optimal procedure. The collected prizes associated with the corresponding feasible routes are listed at columns (4), (6) and (8) respectively. The expended time necessary in each one of the three tested versions are described in the figure 5.3. Boldface values in each line correspond to the best computational results.

(1)	(2)		(3)	(4)		(5)	(6)		(7)	(8)
Problems	**Min Prize**		**GRASP+VNS (2-opt)**	**Collected Prize**		**GRASP+VNS (3-opt)**	**Collected Prize**		**GRASP+VND**	**Collected Prize**
pc50a	400		380,61	573		**380,45**	580		**380,45**	580
pc50b	500		380,61	573		**380,45**	580		**380,45**	580
pc50c	600		382,12	600		**381,38**	600		**381,38**	600
pc50d	700		402,83	706		**394,57**	706		**394,57**	706
pc69a	1000		**672,33**	2763		**672,33**	2763		**672,33**	2763
pc75a	900		506,11	1062		**503,61**	1062		**503,61**	1062
pc75b	1050		504,09	1062		**503,61**	1062		**503,61**	1062
pc100	550		657,71	3596		645,15	3596		**642,53**	3596
pc120a	1000		666,25	1769		**654,03**	1777		**654,03**	1777
pc130	1000		661,62	1876		**653,87**	1887		656,28	1478
pc136	2500		696,01	5773		**685,08**	5787		**685,08**	5787
pc266	5000		1227,11	5008		**1195,13**	5009		1206,87	5017
pc360a	7000		2166,57	7004		2159,32	7001		**2152,59**	7003
pc360b	9000		10.367	9002		**10.190,41**	9001		10227,32	9004

Figure 5.2: Best upper bounds

In all considered versions above, edge-exchange procedures (2 and 3 optimal) as well as the ADD-step algorithm used to define neighborhoods 2 and 3 (node-exchanges) were performed using the best improving strategy. Computational experiments has shown a superior performance of the first improving strategy in the DROP-step algorithm.

Figure 5.4, summarizes the mean objective function values and associated computational times expended at the set of instances at the table of figure 5.1. Different scales were used at both valuations (time and objective function values). It is important to emphasize that, since all results are concerning to a restricted set of instances, carefull must be taken in order to generalize computational results.

(1)		(2)	(3)	(4)
Problems		**GRASP+VNS (2) cpu time (secs)**	**GRASP+VNS (3) cpu time (secs)**	**GRASP+VND cpu time (secs)**
pc50a		**9,31**	19,84	17,51
pc50b		**9,47**	20,74	17,14
pc50c		**10,55**	25,82	21,53
pc50d		**11,17**	40,51	41,30
pc69a		**25,84**	167,89	125,02
pc75a		**39,90**	212,57	169,74
pc75b		**34,59**	215,99	158,91
pc100		**72,55**	1025,49	802,38
pc120a		**131,26**	1333,33	954,03
pc130		**166,11**	2059,68	1503,32
pc136		**177,40**	2398,51	2394,05
pc266		**1875,65**	41.387,35	13927,61
pc360a		**7096,34**	78.460,55	25153,59
pc360b		**6702,83**	113543,83	59471,00

Figure 5.3: CPU time (secs)

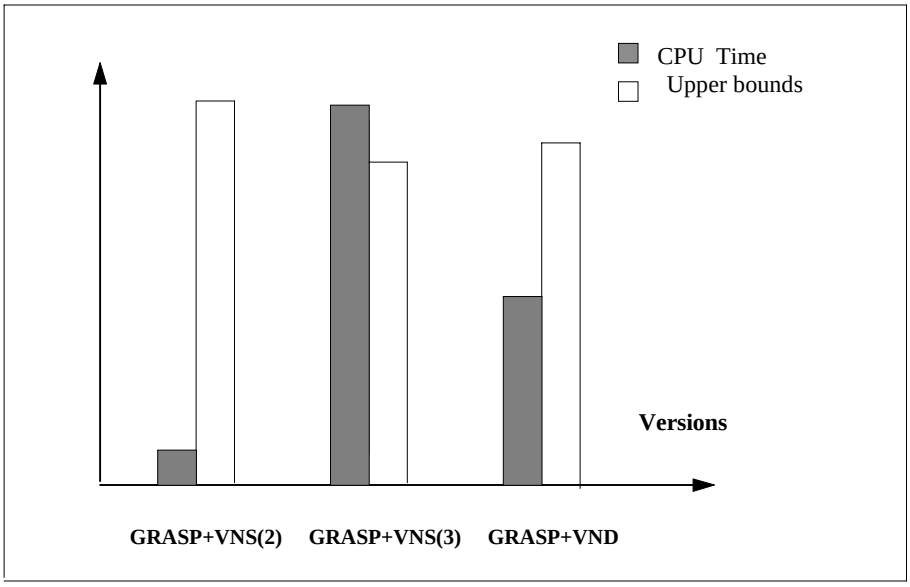

Figure 5.4: Upper bounds and CPU time

As observed in the figures 5.2, 5.3 and 5.4, the GRASP+VNS(2) procedure expends low computational times compared to procedures GRASP+VNS(3) and GRASP+VND respectively. Nevertheless, poor objective function values are obtained. An equilibrium in terms of solution quality (objective function) is observed between GRASP+VNS(3) and GRASP+VND procedures. However, the computational effort expanded by GRASP+VND is a little bit lower than that expended by GRASP+VNS(3) procedure.

Additional tests and reasearch may be applied considering different neighborhood structures in order to try a reduction at the expended computational times. As an example of future research, the GENIUS heuristic proposed by Gendreau et al. [1992] to solve the TSP, can be adapted in a convenient way to define new neighborhoods for the PCTSP. Within the same proposed GRASP+VNS framework, the GENI procedure maybe used in the development of edge-exchange procedures while US algorithm maybe adapted to be applied in the construction of node-exchange procedures.

6 Conclusions

In this paper we present an hybrid metaheuristic that combines Greedy Randomized Adaptive Search Procedure (GRASP) and uses based VNS and VND as local search. A filtering procedure was applied in the GRASP construction phase looking for good starting points to our local seach. The main idea was to combine interesting enhacements of the studied metaheuristics.

Four different neighborhood structures were applied. The edge exchange procedures (2 or 3-optimal) mantains the same set of nodes gathered in the construction phase of GRASP, while at the node-exchange procedures, some node additions and node removals are performed.

The sequence of removals in the second neighborhood mantains feasibility while the third neighborhood allows negative or even infeasible DROP movements remembering strategic oscillation in the tabu search procedure.

Some numerical investigation have been conducted in order to generate upper bounds for the Prize Collecting TSP. Tests considering based VNS and VND as local search were also performed. It is hard to say if the quality of the generated upper bounds pays the substantial increasing in the computional time. Additional research must be done in order to reduce computational effort expended in the local search procedure.

Further research with the PCTSP might be the development of a more evolved exact (e.g. branch-and-bound) algorithm. Logical tests and preprocessing techniques might be helpful in both settings, i.e., when applied to aproximate as well as exact solution procedures. Within the same framework is possible to consider different neighborhood structures. In all cases they must explore increasinly distance neighbours or neighborhoods dealing with different features of the solution space. Another interesting improvements as the reactive GRASP to automaticaly update the size of the Restricted Candidate List and the use of proximate optimality principle, path-relinking or long-term memory can also be considered.

7 References

[01] E. J. Anderson [1996] "Theory and Methodology: Mechanisms for local search", European Journal of Operational Research 88, 139-151.

[02] B. Awebuch; Y. Azar; A. Blum and S. Vempala[1995] " New Approximation Guarantees for Minimum-Weight k-Tress and Prize-Collecting Salesmen", In Proceedings of the 27th Annual ACM Symposium on the Theory of Computing, pp.277-283.

[03] E. Balas, [1989] "The Prize Collecting Traveling Salesman Problem", Networks, Vol.19 pp. 621-636.

[04] E. Balas, [1993] "The Prize Collecting Traveling Salesman Problem: II. Polyhedral results". Technical report MSRR-591, Carnegie Mellon University, Pittsburgh, PA 15213-3890.

[05] E. Balas, G. Martin [1985] "ROLL-A-ROUND: Software package for scheduling the rounds of a rolling mill". *Copyright Balas and Martin Associates, 104 Maple Heights Road, Pittsburgh.*

[06] N. Christofides and S. Eilon [1972] "Algorithms for Large-Scale Traveling Salesman Problems", Operational Res. Quart. 23, 511-518.

[07] Dell'Amico, F. Maffioli, P. Varbrand, [1994] "On Prize-Collecting Tours and the Asymmetric Traveling Salesman Problem", Technical Report N. 53. Dipartimento di Elettronica e Informazione, Politecnico di Milano.

[08] T.A. Feo e Resende [1995] "Greedy Randomized Adaptive Search Procedures", Journal of Global Optimization 6, 109-133.

[09] M. Fischetti and P. Toth [1988] "An additive apprach for the optimal solution of the prize-collecting travelling salesman problem". In Golden, B.L. & Assad A.A. (Eds) *Vehicle Routing: Methods and Studies* (pp. 319-343). Elsevier Science Publishers B. V., North-Holland.

[10] M.R. Garey e D.S. Johnson [1979] “Computers and Intractability: A Guide to theory of NP-Completeness”, Freeman and Company, New York.
[11] M. Geandreau, A. Hertz, G. Laporte [1992] “New insertion and postoptimization procedures for the traveling salesman problem”, Operations Research, Vol. 40 n. 6.
[12] F. Glover [1989], “Tabu Search. Part I”, ORSA J. Comput. 1 pp.190-206.
[13] F. Glover [1990], “Tabu Search. Part II”, ORSA J. Comput. 2 pp.4-32.
[14] M. Goemans and D. Willianson [1992] “A General Approximation Technique for Constrained Forest Problems”, In Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, pages 307-315.
[15] P. Hansen, N. Mladenovic, D. Perez-Brito [1998] “Variable Neighborhood Decomposition Search”, Technical Report, Les Cahiers du GERAD, G-98-53.
[16] E. Horowitz, S. Sahni [1978] “Fundamentals of Computer Algorithms”, Computer Science Press Inc.
[17] M. Laguna and R. Martí [1997] “A GRASP for Coloring Sparse Graphs”, Techinical Report, CO 80309-0419,USA.
[18] S. Lin, [1965] “Computer Solutions of the Traveling Salesman Problem”, Bell System Tech. J. 44, pp. 2245-2269.
[19] N. Mladenovic, P. Hansen [1997] “Variable Neighborhood Search”, Computers Ops. Res. Vol.24, n. 11, pp. 1097-1100.
[20] M. Resende [1998] “Greedy Randomized Adaptive Seach Procedure (GRASP)”, AT&T Labs Research Technical Report: 98.41.1.
[21] S. Voβ [1996] “Dynamic tabu seach strategies for the traveling purchaser problem”Annal of Operations Research 63 pp 253-275.