Hybrid Metaheuristic for the Prize Collecting Travelling Salesman Problem

Antonio Augusto Chaves and Luiz Antonio Nogueira Lorena *

National Institute for Space Research - INPE
Laboratory of Computing and Applied Mathematics
So Jos dos Campos, Brazil
chaves;lorena@lac.inpe.br

Abstract. The Prize Collecting Traveling Salesman Problem (PCTSP) can be associated to a salesman that collects a prize in each city visited and pays a penalty for each city not visited, with travel costs among the cities. The objective is to minimize the sum of travel costs and penalties, while including in the tour enough cities to collect a minimum prize. This paper presents one solution procedure for the PCTSP, using a hybrid metaheuristic known as Clustering Search (CS), whose main idea is to identify promising areas of the search space by generating solutions and clustering them into groups that are them explored further. The validation of the obtained solutions was through the comparison with the results found by CPLEX.

1 Introduction

This paper presents a new hybrid metaheuristic to solve the Prize Collecting Traveling Salesman Problem (PCTSP). The PCTSP is a generalization of the Traveling Salesman Problem (TSP), where a salesman collects a prize p_i in each city visited and pays a penalty γ_i for each city not visited, considering travel costs c_{ij} between the cities. The problem intend to minimize the sum of travel costs and penalties paid, while including in the tour enough cities to collect a minimum prize (p_{min}), defined a priori. In this tour, each city can be visited at most one time.

The solution of PCTSP is difficult due to a large number of possible solutions. Since the PCTSP generalizes the TSP it is also a NP-hard problem, as the TSP is a particular case of PCTSP where the minimum prize is same to the sum of prizes of all nodes.

In this paper, the PCTSP is solved using a hybrid metaheuristic, known as Clustering Search (CS), which was proposed by Oliveira and Lorena [1]. The CS consists of detecting promising areas of the search space using an algorithm that generates solutions to be clustered. These promising areas may then be explored through local search methods as soon as they are discovered. The algorithm used

* The authors acknowledge CNPq by partial research support

to generate the solutions was a method combining Greedy Randomized Adaptive Search Procedure (GRASP) [2], Variable Neighborhood Search (VNS) [3].

The commercial solver CPLEX [4] has been used to solve the formulation of the PCTSP for small size problems, in order to validate the computational results of CS.

The remainder of the paper is organized as follows. Section 2 reviews previous works about PCTSP. Section 3 describes the metaheuristic that was used in this paper, and section 4 present the CS applied to PCTSP. Section 5 presents the computational results and section 6 concludes the paper.

2 Literature Review

The PCTSP was introduced by Egon Balas [5, 6] as a model for scheduling the daily operations of a steel rolling mill. The author presented some structural properties of the problem and two mathematical formulations.

Fischetti and Toth [7] developed several bounding procedures, based on different relaxations. A branch and bound algorithm was also developed that was applied to small size problems.

Goemans and Williamson [8] provided a 2-aproximation procedure to a version of the PCTSP, without the minimum prize constraint. Dell'Amico et al. [9] developed a Lagrangean heuristic, which use a Lagrangean relaxation for generating starting solutions for the heuristic procedure. A procedure Adding-Nodes was used to obtain feasible solutions for PCTSP through the lower bound and a procedure Extension and Collapse seeks improving feasible solutions.

Chaves and Lorena [10] proposed new heuristics based in CS to solve the PCTSP, firstly using an genetic algorithm as generator of solutions for the clustering process, and later changing the genetic algorithm for other metaheuristics.

Feillet et al. [11] presents a survey on TSP with profits that include the PCTSP, identifying and comparing the complexity of different classes of applications, modelling approaches and exact or heuristic solution techniques.

3 Clustering Search

The Clustering Search (CS) generalizes the Evolutionary Clustering Search (ECS) proposed by Oliveira and Lorena [1], that employ clustering for detecting promising areas of the search space (see also [12]). It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. An area can be seen as a search subspace defined by a neighborhood relationship in metaheuristic coding space. In the ECS, a clustering process is executed simultaneously to an evolutionary algorithm, identifying groups of individuals that deserve special interest. In the CS, the evolutionary algorithm was substituted by distinct metaheuristics, such as Simulated Annealing, GRASP, Tabu Search, VNS and others.

The CS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple $\mathcal{G} = \{C; r; \beta\}$ where C, r and β

are, respectively, the center and the radius of the area, and a search strategy associated to the cluster.

The center C is a solution that represents the cluster, identifying the location of the cluster inside of the search space. Initially, the centers are obtained randomly and progressively it tends to slip along really promising points in the close subspace. The radius r establishes the maximum distance, starting from the center, that a solution can be associated to the cluster. For example, in combinatorial optimization, r can be defined as the number of movements needed to change a solution into another. The search strategy β is a systematic search intensification, in which solutions of a cluster interact among themselves along the clustering process, generating new solutions.

The CS consists of four conceptually independent components with different attributions: a search metaheuristic (SM); an iterative clustering (IC); an analyzer module (AM); a local searcher (LS);

The search metaheuristic (SM) component works as a full-time solution generator. The algorithm is executed independently of the remaining components and must to be able provide the continuous generation of solutions directly for the clustering process. Simultaneously, clusters are maintained to represent these solutions. This entire process works like an infinite loop, in which solutions are generated along the iterations.

The iterative clustering (IC) component aims to gather similar solutions into groups, maintaining a representative cluster center for them. To avoid extra computational effort, IC is designed as an online process, in which the clustering is progressively fed by solutions generated in each iteration of SM. A maximum number of clusters $\mathcal{NC}$ is an upper bound value that prevents an unlimited cluster creation. A distance metric must be defined, a priori, allowing a similarity measure for the clustering process.

The analyzer module (AM) component provides an analysis of each cluster, in regular intervals, indicating a probable promising cluster. A cluster density, δ, is a measure that indicates the activity level inside the cluster. For simplicity, δ_i counts the number of solutions generated by SM and allocated to the cluster i. Whenever δ_i reaches a certain threshold, meaning that some information template becomes predominantly generated by SM, such information cluster must be better investigated to accelerate the convergence process on it.

At last, the local search (LS) component is a local search module that provides the exploitation of a supposed promising search area, framed by cluster. This process can happen after AM having discovered a promising cluster and the local search is applied on the center of the cluster. LS can be considered as the particular search strategy β associated with the cluster, i.e., a problem-specific local search to be employed into the cluster.

4 CS for PCTSP

A version of CS for the PCTSP is presented in this section. The application details are now described, clarifying the approach.

4.1 The GRASP/VNS Metaheuristic

The component SM, responsible for generating solutions to the clustering process, was a metaheuristic that combines GRASP and VNS.

The GRASP [2] is 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 phase.

The construction phase of GRASP uses the insertion rule of the procedure Adding-Nodes [9] to build the candidate list (C_{list}). Initially two nodes are added in the tour and then each node is selected in a random way starting from a part of the C_{list} containing the best candidates, called Restricted Candidate List (RCL). This node is added to the solution and the candidate list is updated at each iteration. The construction phase stops when does not exist candidates with positive costs and the minimum prize be collected. The insertion rule for add a node k between the nodes i and j is

$$g(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj} \tag{1}$$

that it is composed by the cost of the arc (i, j), the penalty of the node k and the costs of the arcs (i, k) and (k, j), respectively.

The local search phase of GRASP uses the VNS [3], which is a metaheuristic going on a systematic change of neighborhood within a local search algorithm.

Initially a set of neighborhood structures is defined through random movements. The VNS proposed implement nine neighborhood structures, through the following movements:

- m_1: add one node to the tour;
- m_2: drop one node from the tour;
- m_3: swap position from two nodes of the tour;
- m_4: add two nodes to the tour;
- m_5: drop two nodes from the tour;
- m_6: swap position from four nodes of the tour;
- m_7: add three nodes to the tour;
- m_8: drop three nodes from the tour;
- m_9: swap position from six nodes of the tour;

Starting from the current solution, at each iteration, a randomly neighbor is selected in the k^{th} neighborhood of the incumbent solution. That neighbor is then submitted to some local search method. If the solution obtained is better than the incumbent, update the incumbent and continue the search of the first neighborhood structure. Otherwise, the search continues to the next neighborhood. The VNS stopped when the maximum number of iterations since the last improvement is satisfied.

In this paper we used the method Variable Neighborhood Descent (VND) [3] as a local search of the VNS, and it is composed by three different improvement

methods, that combine two movements: Add-step and Drop-step [13]. The Add-step movement consists in a node addition that provides the best value of the addition function. The Drop-step movement consists in a node removal (from tour) that provides the best value of the removal function. In both movements, if the functions value was positive then the objective function will be better after the movement. The main aspect to be observed is that all moves are executed preserving feasibility.

The improvement methods of the VND are:

- **SeqDrop**: to apply a sequence of Drop-step movements while the objective function value is being decreased;
- **AddDrop**: to apply one Add-step movement and one Drop-step movement;
- **SeqAdd**: to apply a sequence of Add-step movements while the objective function value is being decreased.

Whenever some improvement method obtains a better solution, the VND returns to the first improvement method. The stopping condition of the VND was there not to be more improvements for the solution.

4.2 The Clustering Process

The IC is the CS's core, working as a classifier, keeping in the system only relevant information, and driving the search intensification in the promising search areas. Initially, a maximum number of clusters ($\mathcal{NC}$) is defined a priori. The i^{th} cluster has its own center C_i and a radius r that was identical to the other clusters.

Solutions generated by GRASP/VNS are passed to IC that attempts to group as known information, according to a distance metric. The information activates the closest center C_i (cluster center that minimizes the distance metric), causing some kind of perturbation on it. In this paper, the metric distance was the number of different edges between the solution and the center of the cluster, and a larger number of different edges among them imply in more the dissimilarity.

The perturbation means an assimilation process, in which the center of the cluster is update by the new generated solution. Here, we used the path-relinking method [14], that generates several points (solutions) taken in the path connecting the solution generated by GRASP/VNS and the center of the cluster. Since each point is evaluated by the objective function, the assimilation process itself is an intensification mechanism inside the clusters. The new center C_i is the best evaluated solution sampled in the path.

The AM is executed whenever a solution is assigned to a cluster, verifying if the cluster can be considered promising. A cluster becomes promising when the density δ_i reaches a certain threshold, given for:

$$\delta_i \geq \mathcal{PD}.\frac{\mathcal{NS}}{|Clus|} \tag{2}$$

where, $\mathcal{NS}$ is the number of solutions generated in the interval of analysis of the clusters, $|Clus|$ is the number of clusters, and $\mathcal{PD}$ is the desirable cluster density

beyond the normal density, obtained if $\mathcal{NS}$ was equally divided to all clusters. The center of a promising cluster is improved through the LS.

The LS was implemented by a 2-Opt procedure [15], which seeks to improve the center of the promising cluster. The 2-Opt is based on resequencing of the route always leads to a better solution, since it may possibly decrease travel costs, while leaving prizes and penalties unchanged. It amounts to simply considering the set of nodes currently visited by the route and trying to shorten the length of the route through these nodes. In this paper, the 2-Opt consists in 2-changes over a route, deleting two arcs and replacing them by two other arcs to form a new route. This method continues while there is improvement in the route through this movement.

The whole CS pseudo-code is presented in Figure 1.

```
procedure CS
    { SM component }
    for (number of iterations is not satisfied) do
        {construction phase of GRASP}
        s = ∅
        while (solution not built) do
            compute candidate list (C)
            RCL = C * α
            e = select at random a value of RCL
            s = s ∪ {e}
        end while
        {local search phase of GRASP - VNS}
        k_max = number of neighborhoods
        while (stop condition is not satisfied) do
            k = 1
            while (k ≤ k_max) do
                generate at random s' ∈ N^k(s)
                s'' = apply VND with s' as starting point
                if ( f(s'') < f(s) ) then
                    s = s''
                    k = 1
                else
                    k = k + 1
            end while
        end while

        { IC component }
        calculate the distance of the solution GRASP/VNS (s) and the clusters
        insert the solution in the most similar cluster (C_i)
        apply the assimilation process – path-relinking(s, C_i)

        { AM component }
        verify if the cluster can be considered promising. If so, the LS component is
        applied to it

        { LS component }
        apply the 2-Opt heuristic to the promising cluster
    end for
end procedure
```

Fig. 1. CS pseudo-code

5 Computational Results

The CS was coded in C++ and it was run on a Pentium 4 of 3.00 GHz. The experiments were accomplished with objective of evidencing the flexibility of the method in relation to the algorithm used to feed the clustering process, and also to validate the proposed approach, showing that the clustering search algorithm can be competitive to solved the PCTSP.

There are no available instances for PCTSP in the literature. In this paper, test instances were randomly generated as in [9]. We generated problems with n = (20, 40, 60, 80, 100, 200, 300, 400, 500) vertices, travel costs $c_{ij} \in [1, M]$ with $M \in \{1000, 10,000\}$, prizes $p_i \in [1, 100]$, and penalties $\gamma_i \in [1, N]$ with $N \in \{100, 1000, 10,000\}$. The value of minimum prize (p_{min}) has been generated as with $\sigma \in \{0.2, 0.5, 0.8\}$. These test instances are available in `http://www.lac.inpe.br/~lorena/instancias.html`.

The following parameters values for approach CS were adjusted through several executions and are also based in [12]. The following parameters obtained the best results:

- number of solutions generated at each analysis of the clusters $\mathcal{NS} = 200$;
- maximum number of clusters $\mathcal{NC} = 20$;
- density pressure $\mathcal{PD} = 2.5$;
- percentage of the best elements in the RCL, $\alpha = 0.2$.

The formulation presented in [10] was solved using the solver CPLEX 10.0.1, and the results are presented in following Tables. The CPLEX solved the PCTSP up to 100 nodes, founding the optimal solution in a reasonable execution time for small instances. However, for the instances with 80 nodes, the CPLEX took several hours execution to find the optimal solution, and, for any instances with 100 nodes the CPLEX did not get to close the gap between lower and upper bounds in 100,000 seconds. Beside that, the CPLEX did not get to find a feasible solution for the high instances in 100,000 seconds.

Tables 1-4 give the results for the PCTSP. The entries in the tables are:

- value of parameter σ;
- number of vertices (n) in the original graph;
- the best integer solutions (BI) found by the CPLEX, the Gap and running time (RT) of CPLEX. The values of Gap equal zero define that the optimal has been achieved;
- best solution (BS), average solution (AS), average running time (AT) to find it in the CS and the Deviation (DE), that reflects the relative error of the average solution for the CS, relative to the best found solution by CPLEX or CS, and are calculated by (AS - BS or BI)/(BS or BI) $\times$100; and
- best solution (BS), average solution (AS), average running time (AT) to find it in the GRASP/VNS and de Deviation (DE).

The best solutions found (BS), the averages of solutions (AS) and the running times to find the averages solution (AT) were considered to compare the approaches.

The values in boldface show the best objective function values and execution times for each instance.

Table 1 gives the results for all values of n, σ, $c_{ij} \in [1, 1000]$ and $\gamma_i \in [1, 100]$. One can see that the approach CS has better results in 89% of the tests, has found the optimal solutions for instances up to 60 nodes, and solutions better than the CPLEX for instances with 100 nodes. The running times of CS were very competitive related to the CPLEX. The GRASP/VNS, without the clustering process, has worse results than CS in quality of solutions and deviation. The CS was very robust producing small deviations. The same conclusions can be drawn for instances with $c_{ij} \in [1, 10,000]$ and $\gamma_i \in [1, 1000]$ presented in Table 2.

Table 1. PCTSP: Symmetric random instances, $c_{ij} \in [1, 1000]$; $\gamma_i \in [1, 100]$. Times in seconds.

		CPLEX			CS				GRASP/VNS			
σ	n	BI	Gap	RT	BS	AS	AT	DE	BS	AS	AT	DE
	20	903	0.00	0.80	**903**	**903.0**	**0.03**	**0.00**	903	903.0	0.07	0.00
	40	996	0.00	20.46	**996**	**996.0**	**2.28**	**0.00**	996	996.0	4.41	0.00
	60	1314	0.00	474.94	**1314**	**1314.0**	**13.73**	**0.00**	1314	1321.3	25.36	0.56
	80	**1384**	**0.00**	**26692.93**	1386	1392.8	83.03	0.64	1531	1548.0	215.46	11.85
0.2	100	1514	1.85	100,000.00	**1508**	**1526.4**	**196.10**	**1.22**	1552	1562.3	122.38	3.60
	200	-	-	-	**1816**	**1834.4**	**502.94**	**1.01**	1898	1922.3	464.14	5.86
	300	-	-	-	**2281**	**2313.0**	**1069.11**	**1.40**	2439	2506.7	845.10	9.89
	400	-	-	-	**2504**	**2554.2**	**1212.85**	**2.00**	2671	2691.3	1138.07	7.48
	500	-	-	-	**3233**	**3281.1**	**1355.62**	**1.48**	3382	3401.3	1936.49	5.21
	20	1123	0.00	14.45	**1123**	**1123.0**	**0.30**	**0.00**	1123	1140.0	0.35	1.51
	40	996	0.00	21.84	**996**	**996.0**	**2.34**	**0.00**	996	1008.0	4.26	1.20
	60	1314	0.00	468.51	**1314**	**1314.0**	**11.42**	**0.00**	1314	1339.0	18.00	1.90
	80	**1384**	**0.00**	**32121.21**	1388	1396.6	72.74	0.91	1497	1519.7	296.15	9.80
0.5	100	1514	1.75	100,000.00	**1513**	**1534.6**	**181.00**	**1.43**	1562	1584.3	84.83	4.71
	200	-	-	-	**1816**	**1844.2**	**572.46**	**1.55**	1902	1943.7	485.48	7.03
	300	-	-	-	**2171**	**2250.5**	**1213.63**	**3.66**	2428	2452.1	1127.12	12.94
	400	-	-	-	**2489**	**2579.7**	**1490.49**	**3.64**	2694	2744.3	1048.74	10.26
	500	-	-	-	**3159**	**3200.7**	**1784.58**	**1.32**	3246	3298.4	2216.91	4.41
	20	1354	0.00	8.79	**1354**	**1354.0**	**0.15**	**0.00**	1354	1354	0.40	0.00
	40	1129	0.00	44.69	**1129**	**1137.0**	**2.95**	**0.71**	1156	1186.2	4.67	5.07
	60	1319	0.00	474.80	**1319**	**1344.2**	**17.46**	**1.91**	1379	1387.7	9.28	5.21
	80	**1384**	**0.00**	**27498.29**	1396	1400.2	63.86	1.17	1468	1485.3	218.82	7.32
0.8	100	1575	6.08	100,000.00	**1519**	**1537.6**	**186.54**	**1.22**	1563	1589.0	105.63	4.61
	200	-	-	-	**1768**	**1797.2**	**805.01**	**1.65**	1908	1926.7	630.53	8.97
	300	-	-	-	**2148**	**2213.0**	**1528.29**	**3.03**	2314	2444.0	1211.34	13.78
	400	-	-	-	**2455**	**2494.3**	**1668.90**	**1.60**	2599	2669.3	1536.90	8.73
	500	-	-	-	**3214**	**3324.2**	**1708.36**	**3.42**	3438	3471.6	1842.49	8.02

Table 2. PCTSP: Symmetric random instances, $c_{ij} \in [1, 10,000]$; $\gamma_i \in [1, 1000]$. Times in seconds.

		CPLEX			CS				GRASP/VNS			
σ	n	BI	Gap	RT	BS	AS	AT	DE	BS	AS	AT	DE
	20	11677	0.00	2.65	**11677**	**11677.0**	**0.04**	**0.00**	11677	11667.0	0.09	0.00
	40	10776	0.00	21.97	**10776**	**10776.0**	**3.19**	**0.00**	10776	10896.0	7.23	1.11
	60	**14236**	**0.00**	**1151.37**	14243	14314.1	7.11	0.55	14684	15033.2	41.46	5.60
	80	**14484**	**0.00**	**68464.35**	14609	14760.9	114.86	1.91	15022	15327.1	134.18	5.82
0.2	100	14841	10.79	100,000.00	**13620**	**14015.0**	**104.17**	**2.90**	14328	14510.9	79.97	6.54
	200	-	-	-	**15303**	**15628.2**	**528.53**	**2.13**	16250	16560.3	412.80	8.22
	300	-	-	-	**21869**	**22158.0**	**662.13**	**1.32**	22760	23898.2	653.38	9.28
	400	-	-	-	**24390**	**25099.6**	**1354.40**	**2.91**	25685	26639.7	1215.72	9.22
	500	-	-	-	**31090**	**31558.7**	**1643.15**	**1.51**	32965	33666.3	1323.48	8.29
	20	12900	0.00	9.84	**12900**	**12900.0**	**0.17**	**0.00**	12900	12996.9	0.23	0.75
	40	10776	0.00	21.95	**10776**	**10776.0**	**5.58**	**0.00**	10776	10861.1	8.03	0.79
	60	**14236**	**0.00**	**1152.99**	14349	14421.9	13.73	1.31	15005	15065.3	51.12	5.82
	80	**14484**	**0.00**	**68464.35**	14512	14830.0	111.92	2.39	15458	15744.9	208.18	8.71
0.5	100	14841	10.79	100,000.00	**13900**	**14089.1**	**118.35**	**1.36**	14447	14696.0	61.04	5.73
	200	-	-	-	**15190**	**15440.4**	**664.17**	**1.64**	16132	16676.3	502.56	9.78
	300	-	-	-	**22731**	**23211.7**	**696.75**	**2.11**	23855	24504.3	840.55	7.80
	400	-	-	-	**23898**	**24525.3**	**1755.43**	**2.63**	25233	25494.1	1395.65	6.68
	500	-	-	-	**30275**	**30842.0**	**2173.11**	**1.87**	32932	33669.3	1532.14	11.21
	20	16559	0.00	6.28	**16559**	**16559.0**	**0.07**	**0.00**	12559	12559.0	0.12	0.00
	40	10776	0.00	31.32	**10776**	**10776.0**	**4.66**	**0.00**	10776	10844.0	5.83	0.63
	60	**14864**	**0.00**	**18508.17**	14864	15017.1	32.74	1.03	15215	15888.2	65.40	6.89
	80	**14484**	**0.00**	**70205.22**	14740	14793.9	92.77	2.14	15195	15329.9	142.87	5.84
0.8	100	17316	23.54	100,000.00	**13704**	**13971.9**	**95.31**	**1.96**	14393	14483.1	104.33	5.68
	200	-	-	-	**15200**	**15376.2**	**716.94**	**1.16**	15891	16249.3	515.52	6.90
	300	-	-	-	**22168**	**22467.7**	**1496.77**	**1.35**	23616	24187.3	1135.30	9.11
	400	-	-	-	**22790**	**23688.3**	**1953.06**	**3.94**	26059	26327.4	1493.63	15.52
	500	-	-	-	**30385**	**30707.0**	**2336.85**	**1.06**	32608	33273.1	1527.11	9.50

The results of Table 3 refer to instances with $c_{ij} \in [1, 10,000]$ and $\gamma_i \in [1, 100]$. The results for these instances were worse than the results for other instances, and, the CS did not get to find the optimal solutions for instances with 60, 80 and 100 nodes, but the solutions were closer to optimal. The deviations were also larger than the others instances. The GRASP/VNS behavior is the probably cause of bad results for these instances.

Table 4 reports the results of computational experiments with $c_{ij} \in [1, 1000]$ and $\gamma_i \in [1, 10,000]$. In this case, the CS has better results in most tests, having found the optimal solutions for instances up to 60 nodes. The running times of the CS were very competitive related to the CPLEX and again the CS was very better than the GRASP/VNS without the clustering process.

Table 3. PCTSP: Symmetric random instances, $c_{ij} \in [1, 10, 000]$; $\gamma_i \in [1, 100]$. Times in seconds.

		CPLEX			CS				GRASP/VNS			
σ	n	BI	Gap	RT	BS	AS	AT	DE	BS	AS	AT	DE
	20	3011	0.00	9.77	**3011**	**3011.0**	**0.01**	**0.00**	3011	3046.2	0.05	1.17
	40	3506	0.00	26.06	**3506**	**3531.6**	**0.67**	**0.73**	3506	3544,4	2.21	1.10
	60	**4251**	**0.00**	**130.31**	4277	4287.2	14.86	0.85	4617	4693.2	9.91	10.40
	80	**4903**	**0.00**	**2226.51**	4903	5044.3	22.61	2.88	5347	5543.3	22.36	13.05
0.2	100	**5635**	**0.00**	**10824.22**	5702	5802.0	59.71	2.96	6095	6125.7	40.44	8.71
	200	-	-	-	**9035**	**9129.0**	**303.41**	**1.04**	9669	9865.3	246.28	9.19
	300	-	-	-	**14592**	**14875.2**	**319.14**	**1.94**	15380	15540.0	444.85	6.50
	400	-	-	-	**16651**	**16850.3**	**640.17**	**1.20**	17297	17564.7	567.84	5.49
	500	-	-	-	**20612**	**21305.7**	**836.**68	**3.37**	21986	22231.0	730.29	7.85
				-								
	20	4313	0.00	7.97	**4313**	**4313.0**	**0.17**	**0.00**	4313	4652.1	0.28	7.86
	40	4694	0.00	59.86	**4694**	**4694.0**	**4.29**	**0.00**	4694	4736.6	8.82	0.91
	60	**6120**	**0.00**	**700.35**	6232	6361.7	14.05	3.95	6739	6937.1	10.57	13.35
	80	**6319**	**0.00**	**72518.87**	6528	6628.1	63.69	4.89	7180	7303.7	88.78	15.58
0.5	100	**6869**	**0.00**	**69562.82**	7710	7833.7	119.52	14.04	8206	8655.0	120.07	26.00
	200	-	-	-	**10293**	**10578.0**	**438.80**	**2.76**	11165	11337.0	231.34	0.51
	300	-	-	-	**15312**	**15698.3**	**549.65**	**2.52**	16872	17646.1	664.34	15.24
	400	-	-	-	**17263**	**17535.7**	**841.70**	**1.58**	18256	18552.3	744.61	7.47
	500	-	-	-	**20896**	**21623.7**	**1056.08**	**3.48**	22136	22300.9	801.02	6.72
				-								
	20	7797	0.00	14.47	**7797**	**7797.0**	**0.11**	**0.00**	7797	7958.6	0.22	2.07
	40	9070	0.00	43.26	**9070**	**9171.6**	**7.55**	**1.12**	9224	9302.6	7.27	2.56
	60	**9459**	**0.00**	**10854.96**	9664	9810.5	32.77	3.72	9934	10165.1	22.20	7.46
	80	**9699**	**0.00**	**98073.17**	9991	10048.0	97.02	3.60	11118	11315.2	108.70	16.96
0.8	100	**10002**	**0.31**	**100,000.00**	10641	10724.1	157.96	7.22	11267	11470.0	131.20	14.68
	200	-	-	-	**12650**	**13024.0**	**616.84**	**2.96**	13448	13853.1	406.51	9.51
	300	-	-	-	**18253**	**18740.8**	**1100.30**	**2.67**	19532	20128.3	908.09	10.27
	400	-	-	-	**18501**	**18955.7**	**1574.56**	**2.46**	19996	21119.3	1027.42	14.15
	500	-	-	-	**23234**	**23590.3**	**1920.92**	**1.53**	24239	25364.9	1451.63	9.17

6 Conclusions

This paper presented the Clustering Search (CS) approach to solve the PCTSP. The CS uses the concept of hybrid algorithms, combining metaheuristics with a clustering process, detecting promising search areas (clusters). Whenever an area is considered promising some aggressive search strategy is accomplished in this area.

The CS is a generalization of the Evolutionary Clustering Search (ECS) [1]. The ECS is a new method that obtained success in unconstrained continuous optimization and has been applied to some combinatorial optimization problems found in the literature, such as pattern sequencing problems. The evolutionary component can be substituted by other metaheuristics and this paper tested the GRASP/VNS procedure to generate solutions to be clusterized.

The results obtained by CS for PCTSP were more competitive than the CPLEX, getting to find the optimal solutions for instances up to 60 nodes.

Table 4. PCTSP: Symmetric random instances, $c_{ij} \in [1, 1000]$; $\gamma_i \in [1, 10, 000]$. Times in seconds.

		CPLEX			CS				GRASP/VNS			
σ	n	BI	Gap	RT	BS	AS	AT	DE	BS	AS	AT	DE
	20	1192	0.00	1.21	**1192**	**1192.0**	**0.23**	**0.00**	1192	1212.7	0.17	1.73
	40	1449	0.00	180.33	**1449**	**1449.0**	**6.52**	**0.00**	1449	1506.2	5.93	3.95
	60	1666	0.00	173.17	**1666**	**1670.2**	**34.94**	**0.25**	1687	1702.7	30.85	2.20
	80	**1794**	**0.00**	**41884.95**	1807	1814.6	144.96	1.15	1842	1846.3	136.71	2.92
0.2	100	**1601**	**0.00**	**22767.73**	1628	1647.0	281.07	2.87	1680	1696.3	189.99	5.95
	200	-	-	-	**1898**	**1946.8**	**911.18**	**2.57**	2060	2091.3	481.61	10.19
	300	-	-	-	**2246**	**2334.0**	**920.96**	**3.92**	2462	2501.0	969.01	9.37
	400	-	-	-	**2880**	**2933.7**	**1370.23**	**1.86**	3049	3105.3	1455.80	7.82
	500	-	-	-	**3385**	**3428.5**	**2455.34**	**1.29**	3491	3688.1	2039.97	8.95
	20	1192	0.00	1.22	**1192**	**1192.0**	**0.24**	**0.00**	1192	1218.7	0.20	2.24
	40	1449	0.00	180.12	**1449**	**1449.0**	**6.05**	**0.00**	1449	1544.8	8.43	6.61
	60	1666	0.00	172.99	**1666**	**1670.8**	**36.33**	**0.29**	1684	1690.0	33.03	1.44
	80	**1794**	**0.00**	**32442.24**	1813	1821.6	131.57	1.54	1855	1876.1	112.92	4.57
0.5	100	**1601**	**0.00**	**22730.51**	1655	1665.3	220.50	4.02	1695	1711.3	141.33	6.89
	200	-	-	-	**1968**	**2012.0**	**529.79**	**2.24**	2052	2087.7	516.25	6.08
	300	-	-	-	**2300**	**2382.3**	**1152.44**	**3.58**	2447	2484.3	1118.08	8.01
	400	-	-	-	**2842**	**2935.3**	**1348.65**	**3.28**	3063	3123.3	1496.11	9.90
	500	-	-	-	**3274**	**3332.6**	**2054.47**	**1.79**	3500	3587.4	1878.52	9.57
	20	1192	0.00	1.21	**1192**	**1192.0**	**0.25**	**0.00**	1192	1230.7	0.23	2.40
	40	1449	0.00	180.55	**1449**	**1449.0**	**14.07**	**0.00**	1449	1534.8	9.80	5.92
	60	1666	0.00	172.85	**1666**	**1669.6**	**46.41**	**0.22**	1687	1691.3	30.85	1.52
	80	**1794**	**0.00**	**27699.33**	1801	1815.2	110.43	1.18	1863	1868.0	83.50	4.12
0.8	100	**1601**	**0.00**	**22722.64**	1626	1658.8	235.88	3.61	1700	1712.3	193.64	6.95
	200	-	-	-	**1978**	**1999.8**	**609.86**	**1.10**	2050	2062.3	665.13	4.26
	300	-	-	-	**2319**	**2371.7**	**1100.99**	**2.27**	2473	2536.2	969.01	9.37
	400	-	-	-	**2837**	**2849.0**	**1554.54**	**0.42**	2876	2907.3	1864.01	2.48
	500	-	-	-	**3305**	**3333.2**	**2134.51**	**0.85**	3491	3494.5	1985.14	5.73

Besides, the CS obtained better results than CPLEX for any instances with 100 nodes and was better than the GRASP/VNS alone. The running times of CS were reasonably small considering the complexity of the PCTSP.

Further work can be proceeded testing the integration of new algorithms on CS, like other metaheuristics such as Ant Colony System, Tabu Search, or Simulated Annealing, and apply the CS in other generalizations of the TSP, such as Profitable Tour Problem (PTP) and Quota TSP.

References

1. Oliveira, A.C.M., Lorena, L.A.N.: Detecting promising areas by evolutionary clustering search. Advances in Artificial Intelligence. Springer Lecture Notes in

Artificial Intelligence Series (2004) 385–394
2. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
3. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research **24** (1997) 1097–1100
4. ILOG France: ILOG CPLEX 10.0 - User's Manual. (2006)
5. Balas, E.: The prize collecting travelling salesman problem. In: Anais..., ORSA/TIMS Meeting (1986)
6. Balas, E.: The prize collecting travelling salesman problem. Networks **19** (1989) 621–636
7. Fischetti, M., Toth, P.: An additive approach for the optimal solution of the prize collecting traveling salesman problem. Vehicle Routing: Methods and Studies (1988) 319–343
8. Goemans, M.X., Williamson, D.P.: A general aproximation technique for constrained forest problems. SIAM Journal on Computing **24**(2) (1995) 296–317
9. Dell'Amico, M., Maffioli, F., Sciomanchen, A.: A lagrangian heuristic for the prize collecting travelling salesman problem. Operations Research **81** (1998) 289–305
10. Chaves, A.A., Lorena, L.A.N.: Hybrid algorithms with detection of promising areas for the prize collecting traveling salesman problem. In: Proceedings..., Los Alamitos, California, International Conference on Hybrid Intelligent Systems, IEEE Computer Society (2005) 49–54
11. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science **2**(39) (2005) 188–205
12. Oliveira, A.C.M., Lorena, L.A.N.: Hybrid evolutionary algorithms and clustering search. In Grosan, C., Abraham, A., Ishibuchi, H., eds.: Hybrid Evolutionary Systems - Studies in Computational Intelligence. Springer SCI Series (2007) 81–102
13. Gomes, L.M., Diniz, V.B., Martinhon, C.A.: An hybrid grasp+vnd metaheuristic fo the prize collecting traveling salesman problem. Simpósio Brasileiro de pesquisa Operacional (SBPO) **32** (2000) 1657–1665
14. Glover, F.: Tabu search and adaptive memory programing: Advances, applications and challenges. Interfaces in Computer Science and Operations Research (1996) 1–75
15. Croes, G.: A method for solving travelling salesman problems. Operations Research **6** (1958) 791–812