Hybrid Algorithms with Detection of Promising Areas for the Prize Collecting Travelling Salesman Problem

Antonio Augusto Chaves
National Institute for Space Research (INPE)
São José dos Campos, Brazil
chaves@lac.inpe.br

Luiz Antonio Nogueira Lorena
National Institute for Space Research (INPE)
São José dos Campos, Brazil
lorena@lac.inpe.br

Abstract

The Prize Collecting Travelling Salesman Problem (PCTSP) is a generalization of the Travelling Salesman Problem. It 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 the costs of the trip and penalties, including in the tour an enough number of cities that allow collecting a minimum prize. This paper approaches new heuristics to solve the PCTSP, using a hybrid evolutionary algorithm, called Evolutionary Clustering Search (ECS) and an adaptation of this, called *CS, where the evolutionary component will be substituted by the metaheuristics GRASP and VNS. The validation of the obtained solutions will be through the comparison with the results found by a commercial solver that was able to solve only small size problems.

1 Introduction

This paper presents a new hybrid heuristic to solve the Prize Collecting Travelling Salesman Problem (PCTSP). The PCTSP is a generalization of the Travelling Salesman Problem, 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 an enough number of cities that allow collecting a minimum prize (p_{min}), defined a priori.

The PCTSP is of difficult solution due the large number of possible solutions. Admitting that the travel costs between the cities are symmetrical, the total number of possible solutions is $(n-1)!/2$, being classified in the literature as NP-hard.

The PCTSP was introduced by Balas [2] 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. Bounding procedures, based on different relaxations, were developed by Fischetti and Toth [7] and Dell'Amico et al. [5].

Goemans and Williamson [9] provide a 2-aproximation procedure to a version of the PCTSP, in which the minimum prize to be collected is removed.

Gomes et al. [10] and Melo and Martinhon [12] present hybrid metaheuristics to solve the PCTSP. The first combines GRASP and VND and the second combines GRASP and VNS as a local search.

Torres and Brito [17] present a new mathematical formulation for PCTSP based on the formulation presented in [2]. In this formulation a new group of constraints is proposed to prevent sub-tours.

Chaves et al. [3] explored two approaches. A mathematical model to PCTSP solved for small instances, and a heuristic procedure, combining the metaheuristics GRASP and VNS/VND.

In this paper, the PCTSP is solved using hybrid heuristics, proposed by Oliveira and Lorena [14], and called Evolutionary Clustering Search (ECS). The ECS consists in detecting promising areas of the search space using an evolutionary algorithm that generates solutions to be clustered. These promising areas should be explored through local search methods as soon as they are discovered. An alternative combination for the clustering search is to substitute the evolutionary algorithm by distinct metaheuristics, such as Greedy Randomized Adaptive Search Procedure (GRASP) [6] and Variable Neighborhood Search (VNS) [13], creating a new hybrid approach that will be called *CS.

The paper also explores a mathematical formulation for the PCTSP, based in Balas [2] and Torres and Brito [17], to validate the computational results of ECS and *CS. The software CPLEX [1] is used to solve this formulation for small size problems.

The remainder of the paper is organized as follows.

Section 2 presents a mathematical model for PCTSP. Section 3 describes the basic ideas and conceptual components of ECS, and sections 4 and 5 present the ECS and *CS applied to PCTSP. Section 6 presents the computational results and section 7 describe some conclusions of this paper.

2 Mathematical model

More formally, the PCTSP can be defined as follows. A weighted graph $G = (V, A)$ is given, where V is the set of nodes of size n and A is the set of arcs of size m. Let us suppose that node 0 is the depot or home city of the salesman. A cost c_{ij} is associated with each arc $(i, j) \in A$, and a prize p_i and a penalty γ_i are associated with each node $i \in V$. Node 0 is then such that $p_0 = 0$ e $\gamma_0 = \infty$.

The mathematical model presented in this paper is based in the formulations proposed by Balas [2] and Torres and Brito [17]. Introducing the binary variables x_{ij} = 1 if arc (i, j) belongs to solution and 0 otherwise, x_{ii} = 0 if node i is visited and 1 otherwise. The PCTSP can be formulated as an integer linear programming problem as follows.

$$\min \sum_{i \in V} \sum_{j \in V} b_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i \in V} p_i x_{ii} \leq \left(\sum_{i \in V} p_i \right) - p_{\min} \tag{4}$$

$$\sum_{j \in V \setminus \{0\}} f_{0j} = 0 \tag{5}$$

$$\sum_{j \in V \setminus \{i\}} f_{ij} = \sum_{j \in V \setminus \{i\}} f_{ji} + p_i x_{ii} \qquad \forall i \in V \setminus \{0\} \tag{6}$$

$$\sum_{j \in V \setminus \{0\}} f_{j0} = \sum_{j \in V \setminus \{0\}} p_j x_{jj} \tag{7}$$

$$f_{ij} > x_{ij} - 1 \qquad \forall i \in V \setminus \{0\}, \ \forall j \in V, \ i \neq j \tag{8}$$

$$\left(\sum_{j \in V} p_j \right) x_{ij} \geq f_{ij} \qquad \forall i \in V \setminus \{0\}, \ \forall j \in V \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{10}$$

$$f_{ij} \geq 0 \qquad \forall i, j \in V \tag{11}$$

where $b_{ij} = \begin{cases} c_{ij}, \text{ if } i \neq j \\ \gamma_i, \text{ if } i = j \end{cases}$

The objective function (1) minimizes the travel costs of arcs included in the tour and penalties to be paid for not visited nodes. Constraints (2) and (3) are the well-known assignment constraints. Constraint (4) is a knapsack-like constraint and ensures that the total collected prize is greater than the minimum prize. Constraints (5) - (9) forces every node visited to be connected to the depot, and give the so-called sub-tour elimination constraints. The flow variables f_{ij} are used to prevent sub-tours assigning the price of the visited node to the edge leaving this node. Finally (10) and (11) are the constraints on the variables of the problem. For resolution of this model, the software CPLEX [1], version 7.5, was used looking for optimal solutions to PCTSP. The CPLEX was successful only for instances up to 51 nodes.

3 Evolutionary Clustering Search

The Evolutionary Clustering Search (ECS) is an evolutionary technique proposed by Oliveira and Lorena [14] that employs clustering to detect promising areas of the search space. It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. In the ECS, a clustering process is executed simultaneously to an evolutionary algorithm, identifying groups of individuals that deserve special interest.

The ECS tries to locate promising areas through the framing of these for clusters. A cluster is defined by a tuple $G = \{c; r; s\}$ where c, r and s are, respectively, the center and the radius of the area, and a search strategy to be associated to the cluster.

The center is an individual that represents the cluster, identifying the location of the cluster inside of the search space. The radius establishes the maximum distance, starting from the center, that an individual can be associated to the cluster. The search strategy is a systematic search intensification, in which individuals of a cluster interact among themselves, along the clustering process, generating new individuals.

The ECS consists of four conceptually independent components with different attributions:

- an evolutionary algorithm (EA);
- an iterative clustering (IC);
- an analyzer module (AM);
- a local searcher (LS);

Figure 1 shows the four components, the population and the clusters of interacting individuals.

The EA component works as a full-time solution generator. The population evolves independently of the remaining components. Individuals are selected, crossed over, and updated for the next generations. Simultaneously, clusters are maintained to represent these individuals.

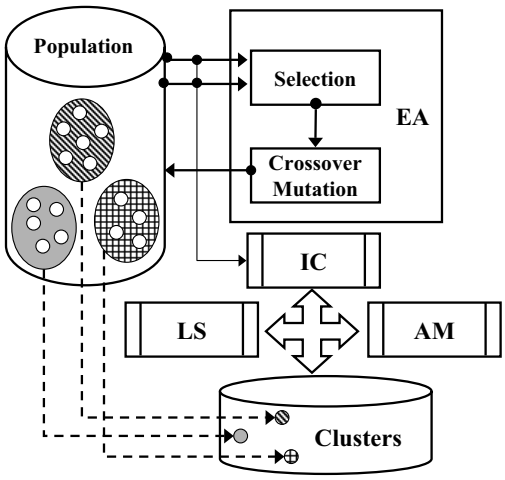

Figure 1. ECS components.

The IC component is the kernel of ECS, working as a classifier of information (solutions represented by individuals) into groups, maintaining in the system just information that is relevant for the process of search intensification. IC is designed as an iterative process that forms groups by reading the individuals being selected or updated by EA.

The AM component provides an analysis of each cluster, in regular intervals of generations, indicating a probable promising cluster. Typically, the density of the cluster is used in this analysis, that is, the number of selections or updating that happened recently in the cluster. A cluster with high density should have a promising center. AM is also responsible for the elimination of clusters with lower densities.

Finally, the component LS is a local search module that provides the exploration of a supposed promising search area. This process happens after the component AM has discovered a promising cluster. The local search is applied on center of the cluster.

4 ECS for PCTSP

We describe in the following the ECS application to PCTSP.

The component EA, responsible for generating solutions to clustering process, will be the Population Training Algorithm (PTA) [15] employing well-known genetic operators as the base-guide selection [11], the crossover BOX [16] and the mutation 2-Opt.

An individual will be represented through a vector that contains the nodes of the problem in the order in that they are visited. Observe that negative signs indicate not visited nodes. The individual representation is shown in the Figure 2, where the sequence of visits is $\{1, 3, 0, 4\}$ and the nodes 5 and 2 were not visited.

1	3	-5	0	4	-2

Figure 2. Individual representation.

The implementation details are now described. The PTA works with a dynamic population of individuals, and, initially the population is randomly generated. All individuals are evaluated by two functions, f and g. The first evaluates the quality of individual and the second applies a problem-specific heuristic (called training heuristic) to evaluate the neighborhood of individual, being the value of the best solution found attributed to g.

In this paper, the training heuristics used to determine the characteristics wanted in the training along the evolutionary process of PTA is the method SeqDrop-SeqAdd [10], that consists of applying a sequence of node removal while the objective function value is being decreased and a sequence of node additions while some improvement is attained (see Figure 3).

In each generation a constant number of individuals (NS) are selected. Two individuals are selected for each crossover, which produces only one new offspring. This offspring can, eventually, suffer mutation.

The adaptation of an individual is proportional to its ranking δ,

$$\delta = d.\,[G_{\max} - g] - |f - g| \tag{12}$$

that is composed by:

- a component concerning the adaptation of individual in relation to the training heuristic: minimizing $(f - g)$;
- a component that privileges the minimization of the function g, thought the minimization of a distance between the individual and an estimate of an upper bound for all the possible values that the functions f and g can assume: the constant G_{max};
- and a constant d, $0 \leq d \leq 1$, to balance the two components of equation 12;

So, better individuals have greater ranks.

The population is then controlled in a dynamic way by an adaptive rejection threshold, τ,

$$\tau_{i+1} = \tau_i + \xi.\,|P|\,.\frac{(\delta_1 - \delta_{|P|})}{RG} \tag{13}$$

that is updated during the evolutionary process. Expression 13 uses the current population size, $|P|$, the best (δ_1) and worst $(\delta_{|P|})$ rankings of individuals in current population, the estimated remaining number of generations, RG, and the ξ constant that controls the speed of the evolutionary

process. At the end of each generation the individuals less adapted ($\delta \leq \tau$) are eliminated from the population.

The ECS component IC executes an iterative clustering of each selected individual. Initially, a maximum number of clusters (MC) is defined. The i^{th} cluster has its own center c_i and a radius r that was identical to the other clusters.

Whenever a selected individual is far away from all centers, a new cluster must be created. Otherwise, the individual should cause a disturbance (assimilation) in the most similar center. In this paper, the metric 2-change was used to calculate the distance among two solutions. This metric computes the amount of necessary changes of two nodes in PTA representation, to transform the selected individual in the center of the cluster.

In the assimilation process uses path-relinking method [8], which accomplishes exploratory movements in the path that connects the selected individual and the center of the cluster. Therefore, the assimilation process is already a method of local search inside of the cluster.

The ECS component AM is executed whenever an individual is assigned to a cluster, verifying if the cluster can be considered promising. A cluster becomes promising when reaches a certain *density* λ,

$$\lambda_t \geq PD.\frac{NS}{|C_t|} \tag{14}$$

where, PD is the desirable cluster density beyond the normal density, obtained if NS was equally divided to all clusters, and $|C_t|$ is the cardinality of cluster t. The center of a promising cluster is refined through the component LS.

The component AM also has as function of cooling all clusters that were activated in each generation, decreasing the density of the clusters. It is also used to eliminate clusters with low density.

The ECS component LS was implemented by the 2-Opt heuristic [4]. This method works trying to find better centers (solutions), through changes of two arcs.

5 *CS for PCTSP

Another approach proposed to solve PCTSP is the *CS, that intends to substitute the component EA for alternative metaheuristics capable to generate a large number of different solutions for the clustering process. In this paper we propose to use a hybrid metaheuristic that combines GRASP [6] and VNS [13].

The GRASP, Feo and Resende [6], 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 used the insertion rule of the procedure Adding-Nodes [5] to build the candidate list (C). Each element is selected in a random way starting from a part of the list C, containing the best candidates, called Restricted Candidate List (RCL). This element is added to the solution and the candidate list is updated at each iteration.

In the local search phase of GRASP uses the VNS, Mladenovic and Hansen [13], 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 six neighborhood structures, through the following movements:

- m_1: inserts 2 nodes in the tour;
- m_2: removes 2 nodes of the tour;
- m_3: changes 4 nodes of the tour;
- m_4: inserts and removes 1 node of the tour;
- m_5: removes 3 nodes of the tour;
- m_6: removes 1 node and changes 4 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. This procedure stops when the time without improvements goes larger than 100 seconds.

The local optimum within a given neighborhood is not necessarily an optimum within other neighborhoods, and a change of neighborhoods can also be performed during the local search phase. This local search is then called Variable Neighborhood Descent (VND) [13].

The implemented VND is composed by three different improvement methods: (1) SeqDrop-SeqAdd [10] (see Figure 3); (2) 2-Opt [4] and (3) Add-Drop [10] (see Figure 4). Whenever some improvement method obtains a better solution, the VND returns to the first improvement method.

The *CS approach has the same components that ECS: iterative clustering (IC), analyzer module (AM) and local searcher (LS). These were implemented in identical way of ECS and are not described again.

6 Computational Results

The ECS and *CS were coded in C++ and were run on *AMD Athlon XP 1.53 GHz* with *256 of RAM Memory*. 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

procedure SeqDrop-SeqAdd
  for (each $k \in R$) **do**
    $L \leftarrow h(k) = c_{p_k k} + c_{k s_k} - c_{p_k s_k} - \gamma_k$,
      such that $h(k) > 0$ and $\left(\sum_{i \in R} p_i\right) - p_k \geq p_{\min}$
  repeat
    Select $k = \max\{h(k)$ for each $k \in L\}$
    Remove k of route R
    Update L
  until $L = \varnothing$

  for (each $k \notin R$) **do**
    $L \leftarrow v(k) = \max_{(i,j) \in A(R)} \{c_{ij} + \gamma_k - c_{ik} - c_{kj}\}$,
      such that $v(k) > 0$
  repeat
    Select $k = \max\{v(k)$ for each $k \in L\}$
    Insert k to route R
    Update L
  until $L = \varnothing$
end procedure

Figure 3. SeqDrop-SeqAdd method.

procedure Add-Drop
  for (each $k \notin R$) **do**
    $L \leftarrow v(k) = \max_{(i,j) \in A(R)} \{c_{ij} + \gamma_k - c_{ik} - c_{kj}\}$
  Select $k = \max\{v(k)$ for each $k \in L\}$
  Insert k to route R

  for (each $k \in R$) **do**
    $L \leftarrow h(k) = c_{p_k k} + c_{k s_k} - c_{p_k s_k} - \gamma_k$
  Select $k = \max\{h(k)$ for each $k \in L\}$
  Remove k of route R
end procedure

Figure 4. Add-Drop method.

to validate the proposed approaches, showing that the clustering search algorithms can be competitive for PCTSP resolution.

There are no available instances for PCTSP in literature. Consequently a set of problems, with $n \in \{11, 21, 31, 51, 101, 251, 501\}$, were randomly generated in the following intervals: travel cost between the nodes: $c_{ij} \in [50, 1000]$; prize associated to each node: $p_i \in [1, 100]$; penalty associated to each node: $\gamma_i \in [1, 750]$. The minimum prize, p_{min}, to be collected represents 75% of the sum of the prizes of all nodes. These test problems are available in http://www.lac.inpe.br/∼lorena/instancias.html.

The following parameters' values for approaches ECS and *CS were adjusted through several executions and are also based in Oliveira and Lorena [14]:

- number of individuals selected at each generation $NS = 200$;
- maximum number of clusters $MC = 20$;
- density pressure $PD = 2.5$;
- upper bound G_{max} is the worst value of an individual in the initial PTA population;
- increment of the rejection threshold $\xi = 0.001$;

The mathematical model presented in section 2 was solved using the software CPLEX 7.5, and the results are presented in the Table 1. The CPLEX solved the PCTSP up to 31 nodes in a reasonable execution time. However, for the larger problems, the CPLEX took several days execution to find the optimal solution. A problem with 101 nodes (*v100a*) was executed giving to CPLEX the feasible solution found in [3] as an upper bound, allowing accelerating the search. Even so, it was executed by three days and did not get to close the gap between lower and upper bounds.

The Table 1 also shows the ECS and *CS results for all test problems. The best solutions found (BS) and the execution time in seconds (ET) were considered to compare the approach performances. The values in bold indicate which approach have better objective function values and execution times for each problem. Note that *CS has found better solutions in all test problems, finding the optimal solution for problems up to 51 nodes.

Table 1. Results of the experiments.

		CPLEX			ECS		*CS	
Problem	$\|V\|$	**BS**	**ET(s)**	**gap**	**BS**	**ET(s)**	**BS**	**ET(s)**
v10	11	1765	0.06	0	1765	0.1	**1765**	**0.05**
v20	21	2302	3.73	0	2302	7.75	**2302**	**0.97**
v30a	31	3582	34.06	0	3582	26.95	**3582**	**1.05**
v30b	31	2515	45.59	0	2515	10.04	**2515**	**1.14**
v30c	31	3236	164.58	0	3236	14.68	**3236**	**1.20**
v50a	51	4328	433439.97	0	4368	210.98	**4328**	**38.40**
v50b	51	3872	241307.43	0	3928	236.30	**3872**	**37.28**
v100a	101	6879	153059.09	2.46	7200	2067.98	**6832**	**719.27**
v250a	251	-	-	-	15935	2958.06	**15162**	**1162.11**
v500a	501	-	-	-	29274	6653.01	**28213**	**2058.37**

Table 2. Average solutions found.

	ECS			*CS		
Problem	**BS**	**AS**	**Rt**	**BS**	**AS**	**Rt**
v10	1765	1765	0	1765	1765	0
v20	2302	2302	0	2302	2302	0
v30a	3582	3582	0	3582	3591	0.24
v30b	2515	2515	0	2515	2515	0
v30c	3236	3236	0	3236	3241	0.14
v50a	4368	4453	1.92	4328	4346	0.42
v50b	3928	3988	1.52	3872	3881	0.24
v100a	7200	7495	3.94	6832	6906	1.07
v250a	15935	16155	1.36	15162	15284	0.80
v500a	29274	30401	3.71	28213	28462	0.87

For each test problem, the ECS and *CS were run 10 times. The Table 2 shows the best solutions found (BS), the averages of solutions found (AS) and the ratio (Rt) between the difference of both and the average solution.

In Table 3, the ECS and *CS are compared against another hybrid metaheuristic, that also combines GRASP and VNS/VND, but do not use clustering search [3]. Once again the *CS seems to better than Chaves et al. [3] in all test problems shown in Table 2.

Table 3. Comparison with another approach.

Problem	Chaves et al.	ET(s)	ECS	ET(s)	*CS	ET(s)
v10	1765	0.1	1765	0.1	**1765**	**0.05**
v20	2302	1.04	2302	7.75	**2302**	**0.97**
v30a	3582	5.43	3582	26.95	**3582**	**1.05**
v30b	2515	3.83	2515	10.04	**2515**	**1.14**
v30c	3236	7.83	3236	14.68	**3236**	**1.20**
v50a	4328	132.45	4368	210.98	**4328**	**38.40**
v50b	3872	43.76	3928	236.30	**3872**	**37.28**
v100a	6892	692.09	7200	2067.98	**6832**	**719.27**
v250a	15310	918.33	15935	2958.06	**15162**	**1162.11**
v500a	28563	2145.79	29274	6653.01	**28213**	**2058.37**

7 Conclusions

This paper proposes two approaches for the resolution of PCTSP, the ECS and *CS. They use the concept of hybrid algorithms, combining metaheuristics with a clustering process, detecting promising search areas. Whenever an area is considered promising some aggressive search strategy is accomplished in this area.

The results obtained by ECS and *CS are competitive for resolution of PCTSP, getting to find the optimal solutions for instances up to 51 nodes. Besides, the *CS approach obtained better results than one taken from the literature that used GRASP and VNS/VND, for the same test problems. These results validate the use of these approaches for the resolution of the PCTSP.

References

[1] *ILOG CPLEX 7.5 Reference Manual*. ©Copyright by ILOG, France, 2001.

[2] E. Balas. The prize collecting travelling salesman problem. *Networks*, 19:621–636, 1989.

[3] A. A. Chaves, F. L. Biajoli, O. M. Mine, and M. J. F. Souza. Modelagens exata e heurística para resolução de uma generalização do problema do caixeiro viajante. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 36:1367–1378, 2004.

[4] G. Croes. A method for solving travelling salesman problems. *Operations Research*, 6:791–812, 1958.

[5] M. Dell'Amico, F. Maffioli, and A. Sciomanchen. A lagrangian heuristic for the prize collecting travelling salesman problem. *Operations Research*, 81:289–305, 1998.

[6] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[7] M. Fischetti and P. Toth. An additive approach for the optimal solution of the prize collecting traveling salesman problem. *Vehicle Routing: Methods and Studies*, pages 319–343, 1988.

[8] F. Glover. Tabu search and adaptive memory programing: Advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, pages 1–75, 1996.

[9] M. X. Goemans and D. P. Williamson. A general aproximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[10] L. M. Gomes, V. B. Diniz, and C. A. Martinhon. An hybrid grasp+vnd metaheuristic fo the prize collecting traveling salesman problem. *Simpósio Brasileiro de pesquisa Operacional (SBPO)*, 32:1657–1665, 2000.

[11] L. Lorena and J. Furtado. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3):309–327, 2001.

[12] V. A. Melo and C. A. Martinhon. Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 36:1295–1306, 2004.

[13] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[14] A. C. M. Oliveira and L. A. N. Lorena. Detecting promising areas by evolutionary clustering search. *Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series*, pages 385–394, 2004.

[15] A. C. M. Oliveira and L. A. N. Lorena. Population training heuristics. *Lecture Notes in Computer Science*, 3448:166–176, 2005.

[16] G. Syswerda. Uniform crossover in genetic algorithms. *International Conference on Genetic Algorithms(ICGA)*, 3:2–9, 1989.

[17] R. D. Torres and J. A. M. Brito. Problemas de coleta de prêmios seletiva. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 35:1359–1371, 2003.