Applied Soft Computing Journal 77 (2019) 261–273

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Improving the Clustering Search heuristic: An application to cartographic labeling

Eliseu J. Araújo [*], Antônio A. Chaves, Luiz A.N. Lorena

Univ. Fed. São Paulo, São José dos Campos, Brazil

HIGHLIGHTS

- The Density Clustering Search is a hybrid method that detects promising areas.
- More efficient clustering techniques are proposed to detect promising regions.
- The results show the hybrid method is competitive for solving optimization problem.

ARTICLE INFO

Article history:
Received 22 March 2017
Received in revised form 7 August 2018
Accepted 2 November 2018
Available online 6 December 2018

Keywords:
Heuristics
Promising regions
Clustering
Labels

ABSTRACT

The use of hybrid metaheuristics is a good approach to improve the quality and efficiency of metaheuristics. This paper presents a hybrid method based on Clustering Search (CS). CS seeks to combine metaheuristics and heuristics for local search, intensifying the search on regions of the search space which are considered promising. We propose a more efficient way to detect promising regions, based on the clustering techniques of Density-based spatial clustering of applications with noise (DBSCAN), Label-propagation (LP), and Natural Group Identification (NGI) algorithms. This proposal is called Density Clustering Search (DCS). To analyze this new approach, we propose to solve a combinatorial optimization problem with many practical applications, the Point Feature Cartographic Label Placement (PFCLP). The PFCLP attempts to locate identifiers (labels) of regions on a map without damaging legibility. The computational tests used instances taken from the literature. The results were satisfactory for clusters made with LP and NGI, presenting better results than the classic CS, which indicates these methods are a good alternative for the improvement of this method.

1. Introduction

Many studies have been done looking for methods to overcome the challenge of finding optimal solutions in optimization problems of practical and/or theoretical application.

Metaheuristic is a method that uses high-level strategies to explore the search space using different heuristics [1]. It has tools to escape from local optima, allowing the search for different promising regions. The challenge is to find, in a short time, solutions close to the optimal solution. Hybrid metaheuristics arose as an alternative to improve the performance of metaheuristics, combining exact methods, heuristics or other metaheuristics.

Applying local search heuristic in all solutions generated by metaheuristics may be impractical regarding the run time. Thus, the selection of a set of solutions to pass through the process is necessary. Therefore, the concept of promising regions can be employed enabling a more rational use of local search and improving the performance of the algorithms. The concept of promising regions can be combined with the use of strategies to identify the positive potential of the search subspace. First, the solutions are identified according to a strategy to find the promising regions. These solutions should then be examined by specific heuristic components.

In this context, [2] developed a hybrid method called Clustering Search (CS). This method is a generic combination of metaheuristics and local search heuristics. The main idea is to detect promising regions of the search space, in which the solutions are produced by metaheuristic, and to group these solutions in clusters that are exploited by local search heuristics. The CS is a flexible framework for solving combinatorial optimization problems.

Instead of applying random or elitist choices of solutions, CS uses intelligent and prioritizing choices to intensify the local search. Therefore, it is expected an improvement in the convergence process and a reduction in the computational effort as a result of the more rational use of heuristics.

[*] Corresponding author.
E-mail addresses: araujo.eliseu@unifesp.br (E.J. Araújo), antonio.chaves@unifesp.br (A.A. Chaves), luizlorena54@gmail.com (L.A.N. Lorena).

https://doi.org/10.1016/j.asoc.2018.11.003
1568-4946/© 2018 Published by Elsevier B.V.

CS was successfully applied to optimization problems, including the capacitated centered clustering problem [3] and the capacitated p-median problem [4].

In this paper we propose a new computationally efficient approach of CS that returns results as good as or better than the current approach. This alternative approach method is named Density Clustering Search (DCS). To this end, we studied three clustering techniques: Density-based spatial clustering of applications with noise (DBSCAN) [5], Label-propagation (LP) [6], and Natural Group Identification (NGI) [7].

DCS was applied to the Point Feature Cartographic Label Placement Problem (PFCLP) [8] with maximal distance to evaluate its performance. PFCLP is a problem that aims to locate cartographic labels on a graphic device keeping the legibility of a given map. This is a recent problem in literature that needs solutions on instances of large-scale data. Computational tests are performed in available instances and the results are compared with the Biased Random-Key Genetic Algorithms (BRKGA) method [9].

The contribution of this work is to improve the grouping process of the CS, seeking to design a more robust and efficient method to detect promising regions and accelerate the exploitation process. Another contribution is to solve larger scale instances of the PFCLP, which mathematical models were not able to solve.

This paper is organized as follows. Section 2 presents related works with CS method. Section 3 presents the algorithms used in the development of this work. Section 4 details the proposed DCS method. Section 5 presents the PFCLP model. Section 6 sets out the implementation of CS, DCS, and BRKGA for the PFCLP. Section 7 presents the results of the computational tests. Section 8 ends with the conclusion and the forthcoming work.

2. CS related works

CS was proposed in [10] being called Evolutionary Clustering Search (ECS). It aims to introduce intelligence and priority to choice of solutions to apply local search heuristics, instead of randomly choosing or applying local search in all solutions. Consequently, an improvement in the convergence process is expected with a decrease in the computational effort, since there is a more rational application of the heuristics. There are different applications in several optimization problems that concluded that CS is efficient and an appropriate option in the search for good solutions.

The first application of CS to a problem was done in [11] on problems of minimization of unrestricted numerical functions. Compared with classical methods, such as Genocop III and OptQuest, CS presented good results.

In another context, the CS was applied to the flow time minimization in permutation flow shop in [12] in order to minimize the sum of the flow times of the tasks. The results were superior for a set of instances in terms of objective function.

In [13] CS was applied to the berth allocation problem. This problem consists of allocating ships to berths using the maximum berth space and minimizing service time. The computational results were compared with methods found in the literature and found better results.

Observing the good results of CS in several studies, [14] have modified the CS for a multiobjective approach and applied it to the container ship loading plan problem, a biobjective problem. The two objectives are minimize the number of movements required to move the containers and the imbalance that the arrangement of the containers may cause during the voyage of the ship. The computational tests were successful, presenting better results than those found in other papers in both objectives.

The success of CS throughout its applications shows it is a method that solves problems from different contexts. The search strategy of promising regions may return solutions with good quality. In this paper we pretend to improve the way that these regions are found. The CS idea of applying exploitation in promising regions was maintained, however the way of detecting is modified.

3. Methods

This section shows details of the CS method and the clustering algorithms used in this work. These methods seek to detect promising regions in the search space of solutions. The promising regions are areas in that metaheuristics generate many solutions with good quality. Clusters represent these regions.

3.1. Clustering search

The CS [2] is a method that detects promising regions (clusters) in the search space of solutions and intensifies the search on these regions. A cluster is defined by one center, c, which is randomly initialized and afterwards tends progressively towards promising points in the search space. The method is divided in four parts: a search metaheuristic (SM), the iterative clustering (IC), the analyzer module (AM), and the local search (LS).

The SM component can be implemented by a metaheuristic that generates diversified solutions on the search space. Usually, it is a full-time solutions generator, exploring the search space by manipulating a set of solutions according to its specific search strategy.

The IC component aims to group solutions in clusters according to a criterion of similarity between solutions. Thus, it is necessary to establish a measure of similarity, Ω, which is a distance metric between the solutions. Therefore, this component aims to maintain similar solutions on the same cluster, and each cluster has a center that represents the solutions belonging to the group.

When the IC component establishes the group and a solution will be grouped, it uses an assimilation process on the center cluster solution c_i and sends a solution sol_k via the SM component. The path-relinking algorithm [15] can be used in this process, producing solutions that combine the features present in c_i and in sol_k. The best solution found in this process is considered the new center of the cluster.

The AM component checks which clusters can be considered promising. These checks are done at intervals of time. Each cluster has an indicator of its volume, ω_j, which intends to show how many solutions it represents. A cluster is considered promising if it reaches a certain number of clustered solutions.

If a cluster is considered promising, its center is sent to a exploitation process (LS component phase). It uses local search heuristics applied to the center cluster solution, exploiting supposed promising search areas.

CS has a simple grouping criterion, which causes certain problems. The lack of treatment of outliers and the presence of many different solutions at the same group are some of the problems. In this paper, we propose a change in the CS grouping method aiming to correct these problems and to provide greater confidence in detecting promising areas. Thus, better results can be achieved by this new method.

The following subsections show the clustering algorithms studied as an alternative to the IC and the AM components of CS. Those are the DBSCAN, the LP, and the NGI algorithms.

3.2. Density-based spatial clustering of applications with noise algorithm (DBSCAN)

The DBSCAN [5] is an algorithm based on density. Its goal is to find algorithms that detect regions of the search space using the density of solutions. DBSCAN was applied with an ant-colony algorithm in the paper of [16] with good results.

DBSCAN assumes that clusters are high-density regions of common patterns between solutions that are separated by low-density regions. Low-density regions are not considered clusters. This algorithm has relevant characteristics. One of them is the treatment of outliers. Outliers are solutions that are not constituent of any cluster. They are considerably different from other solutions with high similarity between them and, therefore, they are separated so that the quality of the groups does not decrease. Before starting the algorithm, it is necessary to set two parameters: *Raio* and *MinPts*. *Raio* is the number that describes a proximity measure by two criteria. The first criterion (*Raio(a)*) relates to the level of similarity of the characteristics between points. Two solutions are considered close if they have a number of equal characteristics. The second criterion (*Raio(b)*) is based on the number of common neighbors. Two solutions can be neighbors if they have a number of common neighbors. *MinPts* is the minimum number of neighbors that a point needs to have to be considered the center of the cluster (*Core point*). Initially, a point *p* is chosen randomly and it is analyzed which points are neighbors of *p* according to the number of common neighbors. Thus, its neighboring points are analyzed according to the number of similar characteristics. If the *MinPts* of *p* is a *Core point*, a cluster is created with *p* as its center. If one neighbor of *p* is considered the *Core point*, a new cluster is created and this neighbor is its center.

3.3. Label-propagation algorithm (LP)

The LP [6,17] is used mainly on detecting network communities [18]. LP has computational complexity linear in $O(m)$ (m is number of edges). LP was chosen to be tested because of the success stories on different types of clustering and, mainly, for its reduced execution time in clustering large amount of data.

Initially, a graph G is formed by solutions of the metaheuristic (SM component). Each vertex $n = 0, \ldots, N\text{-}1$ (N is the total number of vertices) is one solution. The edges of the graph show if two solutions have a level of similarity. Therefore, if a pair of solutions i and j are similar to a certain number of characteristics, they have an edge between them. In most cases, these characteristics are the number of positions with equal values at the vectors representing the solutions of a specific problem. If two solutions represented by vertices i and j have, for example, the same values on 30% of the positions on the vectors solutions, they have an edge between them. This number of similar positions should be set by the user and is a parameter of the algorithm. LP sets different identifications for each vertex of the graph. These identifications can be colors, numbers or names. In this paper, we used numbers. The algorithm proceeds iteratively and reassigns identifications to the vertices so that each vertex has the identification with the higher frequency between its neighbors. This procedure is executed synchronously. If draw occurs, it chooses a random identifier among the draws. The algorithm ends when the identifier of each vertex is the more common in its neighborhood. Each set of solutions that has more than one member and an identical identification between all members is considered a cluster. The best solution of a cluster in terms of objective function is considered the center solution of the cluster.

3.4. Natural Group Identification algorithm (NGI)

The NGI was proposed by [7] and it obtained results with good computational times and achieved the authors goals, identifying the required clusters on different tests. Besides that, NGI offers the possibility of not using parameters.

NGI has the following steps:

1. In the first step, individuals need to be inserted in the process. In this case, the individuals are the solutions of the metaheuristic and a distance matrix of the distances between these solutions;
2. The algorithm starts forming a Spanning Tree (in this paper we used the Kruskal algorithm) and the costs of edges are kept reserved. The edge costs are evaluated by the difference on the solution vectors. A priority queue is formed by costs of edges. The first member of the queue has the higher edge cost of the tree.
3. At iteration, the edges should be removed. The removed edge is the first member of the priority queue. On each removal, a new cluster is formed. After the removal, a function $f(k)$ should be calculated and its values should be reserved. Function $f(k)$ is defined as the added cost of all other edges in all groups minus the number of groups formed.
4. After all removals, a new calculation should be done, now on the values returned by $f(k)$. This calculation refers to the function $B(k)$. $B(k)$ is defined as a norm of $f(k+1) - f(k-1)$. Each $B(k)$ value should be reserved.
5. The lowest value (k) obtained by the difference between $B(k)$ and its precursor $B(k\text{-}1)$ should be found. This k value shows the number of edges to be removed in the Spanning Tree and the number of groups to be made.

The NGI algorithm can use different criteria of edge removal to make natural groups. [7] used the criterion of [19] because of the successful results obtained. This criterion aims to comprise all the possibilities of the groups to be made by the construction of the Spanning Tree and the removal of its edges. It comprises draws formed on a graph made by values of $f(k)$. The ideal removal of edges to make a set of natural groups is the one that induces a change on the graphic gradient, that is, the one that forms a "knee" on the graphic of the function $f(k)$. This "knee" is not seen by the algorithm, so the calculation of $B(k)$ is necessary to find this point on the graph. The minimum value found between $B(k)$–$B(k\text{-}1)$ shows the change on the graphic gradient of the function $f(k)$. Each group made in the end of the process is defined as a cluster. The best solution of a cluster in terms of the objective function is defined as the center solution of the cluster.

4. Density Clustering Search (DCS)

This paper presents a new approach that aims to improve the clustering process of CS with the introduction of new clustering algorithms in the CS method. Fig. 1 shows a conceptual flowchart of our DCS.

The solutions generated by the SM component of DCS are stored in a repository (set of solutions) until a limit *NS*. Periodically, in the IC component, a clustering algorithm is applied on stored solutions to detect clusters of solutions that represent regions of the search space with higher density. The clusters are formed using the clustering criteria of one of the algorithms (DBSCAN, LP or NGI). Each cluster has a central solution that represents the cluster. When a dense cluster is detected, the center of the cluster is sent to the LS component, the local search phase. The process returns to the SM component and this process is repeated until some stopping criterion is reached.

The DBSCAN forms clusters only if it considers that there are high-density patterns in a certain region, so if there is a concentration of solutions in this region, their clusters can be considered promising. The LP looks to dispose of very different solutions from the others (outliers) and has the tendency to form clusters with a high amount of solutions depending on the level of similarity between the solutions. With due rigor, clusters are formed only with high similarity between the solutions of a group. Therefore, these clusters are considered promising. The NGI also seeks to dispose of outliers and has the tendency to form clusters with a considerable amount of solutions. Thus, its clusters were considered promising.

The number of clusters formed is also defined by each algorithm. Therefore, this parameter does not need to be calibrated. In a next phase, a local search (LS component) is applied on the solutions of center clusters to find the best solution for this cluster.

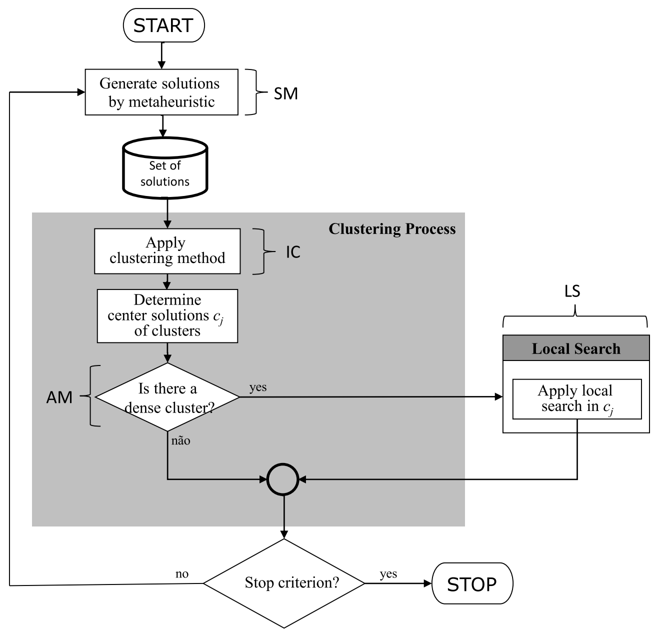

START
Generate solutions by metaheuristic
SM
Set of solutions
Clustering Process
Apply clustering method
IC
Determine center solutions c_j of clusters
AM
Is there a dense cluster?
yes
não
LS
Local Search
Apply local search in c_j
Stop criterion?
no
yes
STOP

Fig. 1. DCS components.
Source: Adapted from [20].

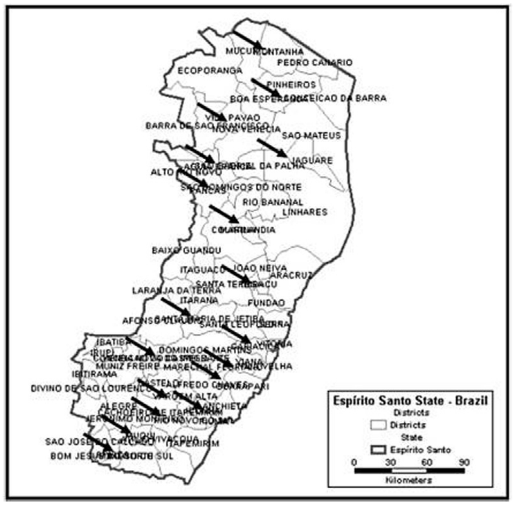


Fig. 2. Example of overlapping of labels [22].

5. Point Feature Cartographic Label Placement (PFCLP)

The cartographic label placement problem refers to the problem of automated label positioning in maps, diagrams, graphs or any graphic object. This is a common problem in Geographic Information Systems (GIS). It is considered that overlapping of labels must be avoided for the sake of legibility [21]. Fig. 2 shows an example of overlapping of labels.

The problem of selecting positions for label placement of point features avoiding overlaps is known in the literature as the PFCLP.

[8] proposed a cartographic standardization for modeling the possible positions of labels, named candidate positions. It uses a priority rank measure designating the best positions, where the smaller number represents the higher priority. Fig. 3 illustrates this standardization considering four candidate positions.

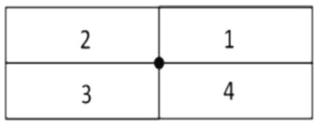


Fig. 3. Cartographic standardization proposed by [8].

There are three main approaches to PFCLP in the literature: the Maximum Independent Vertex Set Problem (MIVSP), the Maximum Number of Conflict Free Labels Problem (MNCFLP) and the Minimum Number of Conflicts Problem (MNCP) [22]. The MIVSP strives to label the maximum number of points avoiding overlaps, and since most of the real problems are too complex, some points may not be labeled with this approach. With the MNCFLP approach, all points must be labeled maximizing the number of free labels, but in general it does not take into account map legibility. The MNCP was proposed as an alternative approach for labeling all points, considering that reducing the total number of conflicts provides a better solution regarding legibility.

There are several studies in the literature that address the three main approaches to PFCLP. Regarding the approach of MIVSP, [23] presented a linear programming formulation and proposed restrictions on the candidates' fictitious positions, carrying penalties of high cost in the objective function. [24] proposed mathematical formulations using cutting restrictions with inequalities for all maximum clicks (complete subgraphs) in the conflict graph. For instance, they applied various heuristics: Simulated Annealing, Diversified Neighborhood Search, and Tabu Search. [25] proposed approximation algorithms, labeling the maximum number of points by computing the set of labels that have not intersect one another.

Regarding the approach of MNCFLP, [8] proposed two algorithms based on a discreet way of descent gradient combined with the Simulated Annealing technique. [26] proposed a Tabu Search algorithm which achieved good results in real instances in a test. An exact algorithm and a Constructive Genetic Algorithm were developed by [27]. The exact algorithm could not achieve good results for instances with more than 25 points, while the genetic algorithm achieved good results for instances of up to 1000 points.

The MNCP was introduced by [22] as a new approach to improve legibility when all points must be labeled and overlaps must be avoided. They presented a mathematical formulation and some Lagrangian relaxation heuristics. [28] proposed a Greedy randomized adaptive search procedure (GRASP) and obtained better solutions than other techniques tested in the literature. [29] presented two mathematical formulations for the MNCP and proposed a Lagrangian with clusters. The main difference between the formulations is how the conflict graph is built: one formulation is based only on the positions of the candidates while the other is based on the positions of the candidates and the points.

There is an interesting and recent work in the literature on PFCLP. [30] present a new way of positioning the labels, performing a positioning calculation in order to achieve a lower number of conflicts. The authors show good results through mapping on maps. The drawback of this approach is the computational cost that is not specified in the paper and the lack of computational tests in available instances to prove its competitiveness.

The interest in working with this problem is that it has been worked on several papers, but with few results for the larger scale contexts, those in which the mathematical models do not find results due to limitations of computational memory. Therefore, this paper presents a metaheuristic method to bring contributions to solve this problem in scales that are not found in mathematical models.

5.1. *New formulation to PFCLP with maximal distance*

In this paper we propose a new formulation for the PFCLP, called MD3, based on dispersion models, solving a problem that combines the ideas of MNCFLP and MNCP.

There are five articles with formulations similar to MD3. [31–34] and [21] deal with PFCLP, but the MD3 formulation differs from them when working with the distance between labels as a penalty given to the objective function. MD3 seeks to provide a larger penalty in the objective function according to the amount of conflicts and also when the distance between conflicting labels is very small. Previously models proposed in the literature do not considerer this feature.

In this formulation, N represents the set of points to be labeled, i the point to be labeled, such that $i \in N$, P_i the set of candidate positions of a point i, j a candidate position, such that $j \in P_i$, x_{ij} a binary variable concerning the decision to choose or not a label, and $x_{ij} = 1$ indicates selecting a candidate position j for the point labeled i. The constant a_{ij} is the cost of cartographic preferences set by [8] and weighs the decision x_{ij} ((i, j) is the pair of indices representative of a point i and label j), S_{ij} is the set of pairs of indices that have conflicts with the pair (i, j), (k, t) is the pair of indices belonging to the set $S_{i,j}$ with $k \in N$, $k > i$ and $t \in P_k$, d_{ijkt} is the distance between the labels (i, j) and (k, t), y_{ijkt} is a binary variable that takes the value 1 if the labels (i, j) and (k, t) have conflict.

The model MD3 is formulated as follows:

$$Max \sum_{i=1}^{n}(\sum_{j=1}^{P_i}(a_{ij}.x_{ij})) - w_i \tag{1}$$

Subject to :

$$\sum_{j=1}^{P_i} x_{ij} = 1 \;\; \forall i = 1 \dots N \tag{2}$$

$$(1/(1 + d_{ijkt}))(x_{ij} + x_{kt} - 1) \le w_i \quad \forall i = 1 \dots N, \; \forall j = 1 \dots P_i, \; (k, t) \in S_{i,j} \tag{3}$$

$$x_{ij}, \; x_{kt} \in \{0, 1\} \quad \forall i = 1 \dots N, \; \forall j = 1 \dots P_i, \; (k, t) \in S_{i,j} \tag{4}$$

$$w_i \ge 0 \;\; \forall i = 1 \dots N \tag{5}$$

The objective function (1) performs the summation of the candidates chosen positions and the costs of these cartographic preferences, and subtracts the conflicts caused by the layout chosen for the labels using the variable w_i. This variable is part of restrictions (3) and aims to assist of inevitable conflicts happening, the conflict of greater distance between labels is chosen for greater dispersion. As w_i is also calculated in the objective function with a negative value, then it is prioritized that w_i should have the smallest possible value so that the objective function can reach its highest value.

This approach aims to decrease the number of overlapping and at the same time decrease the legibility of the map produced by the overlaps that necessarily occur.

Looking at the constraints (3), note that the value $(1/(1 + d_{ijkt}))$ will only be considered in the sum if there is conflict between the labels (i, j) and (k, t). As we seek the smallest possible value for w_i, the greater distances between inevitable conflicts has the priority. Constraints (2) ensure that only one candidate position will be chosen for each point labeled. Constraints (4) and (5) indicate the domain of the model variables.

6. Implementation

In this section we aim to detail how CS, DCS and BRKGA are applied to PFCLP. DCS can be applied to any optimization problem, however, in this work the PFCLP was chosen for tests to verify the efficiency of the method.

1	*2*	*3*	*4*	*5*	*6*	*7*
3	**4**	1	**4**	**2**	**3**	1

Fig. 4. Solution representation to PFCLP.

6.1. *CS and DCS*

CS and DCS have similar structures, but they differ in how the solutions are grouped. Therefore, this subsection explains the components in which they are similar (Search metaheuristic (SM) and Local search (LS)) and which are different (Iterative Clustering (IC) and Analyzer module (AM)).

A vector of labels is used to represent a solution to PFCLP. Each position of the vector represents one point of the map. Each position has a label to represent the point. According to Fig. 3 values of labels varies between 1 and 4. Thus, each position i of the vector can get a value between 1 and 4. Fig. 4 shows an example of the solution representation to PFCLP with seven points.

The goal of PFCLP is to find a solution that minimizes the number of conflicts between labels and also increases the distance between labels in conflict. The objective function of methods is Eq. (6) which is based on Eq. (1). However, the summation of the candidates chosen positions only is done if the candidate position does not have conflicts. The binary variable x_{ij} defines the decision to choose or not a label, $x_{ij} = 1$ indicates that a candidate position j is assigned to the point i. The variable b_{ij} is equal 1 only if the label (i, j) is free. Therefore, it counts the number of free labels. Distances between conflicting labels are used as penalty (d_{ijkt} is the distance between the labels (i, j) and (k, t)). Thus, greater penalties are attributed to smaller distances between conflicting labels.

$$Max \sum_{i=1}^{n}(\sum_{j=1}^{P_i}(b_{ij}.x_{ij})) - (1/(1 + d_{ijkt})) \tag{6}$$

In this paper, Simulated Annealing (SA) metaheuristic [35] is used to generate solutions to CS and DCS. The SA was chosen for this work because it was chosen due to the simple balancing that it performs between diversification of solutions at the beginning of the process and intensification in the search for good solutions to the end.

SA initial solutions and CS initial clusters can influence the final solutions of the results. In [36] was studied the possibility to generate the solutions in a random way and with maximum diversity, in which the solutions are generated in order to obtain a good representation of the solution space. For this work, it was verified that both approaches did not bring significant differences. Therefore the initial solutions of SA and CS are randomly generated.

Three neighborhood structures are used on SA:

- SHIFT: A random position is chosen and a random label replaces the one that was in that position. Fig. 5a shows an example where position 3 is chosen and label 1 is changed to label 3;
- SWAP: Two random positions are chosen and the labels of these positions are interchanged. Fig. 5b shows an example of the application of this structure. Positions 5 and 6 are chosen and labels 3 and 2 are interchanged;
- SHIFT-CONFLICT: In the third structure of SA, one random position is chosen and the labels of the chosen position and of the positions that have conflicts with the chosen position are changed. Fig. 5c shows an example of the application of this structure. Position 3 is chosen and its label, number 1, has a conflict with labels of position 1 (label 2) and position 5 (label 2). Thus, random labels are chosen for all these positions and the position 3 has now the label 2, position 1 has the label 3 and position 5 has the label 3;

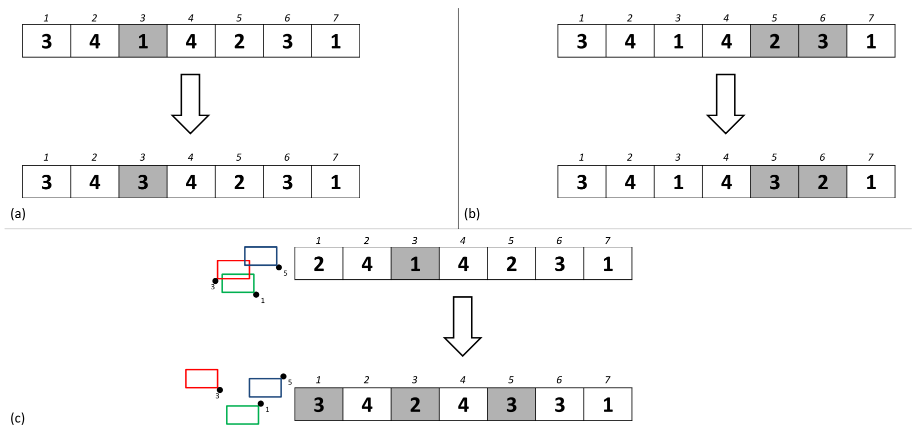


Fig. 5. Neighborhood structures of SA.

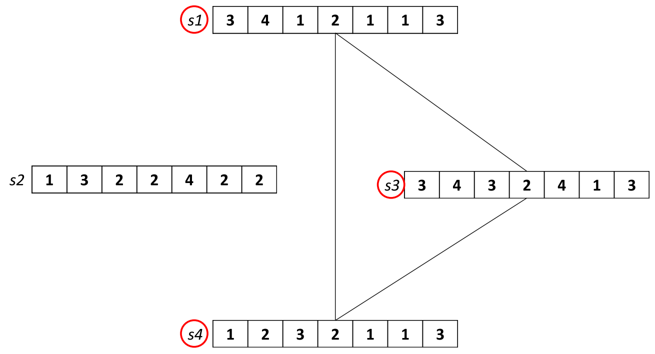


Fig. 6. Example of application of DBScan algorithm.

Solutions formed by SA should then be grouped by CS and by DCS. For this purpose, CS needs a similarity criterion between solutions. DCS needs this same criterion in DBSCAN. In LP, this similarity is used as the criterion to establish edges between solutions on its graph. In NGI, this similarity is necessary to establish edges between solutions and values in the Spanning Tree. Therefore, the vector of labels is used to find similarity measures. The greater the number of identical labels selected for the same point on two solutions of the map, the less is the distance between them. In DCS the solutions of SA are stored in a repository at time intervals to avoid repetition and to increase the diversity. Periodically, the clustering method of DCS is applied on solutions of the repository. In the clustering method of DCS, the algorithms mentioned in this paper were used.

Fig. 6 shows an example of the application of the DBSCAN algorithm. There are 4 solutions *s1*, *s2*, *s3* and *s4*. For two solutions, the neighbors are considered. *Raio(a)* was given the value 4, *Raio(b)* the value of 1 and *MinPts* the value 2. It can be seen that the pairs (*s1*, *s3*) with similarity 5 and 1 neighbor in common, (*s1*, *s4*) with similarity 4 and 1 neighbor in common and (*s3*, *s4*) with similarity 4 and 1 neighbor in common were regarded as neighbors by the value of *Raio(a)*, but they can also be seen surrounding the value of the *Raio(b)*. Each of these solutions having two neighbors were considered the centers of the clusters. Solution *s2* did not fit the criteria and was considered an outlier, so it was discarded.

Fig. 7 shows an example of applying the LP algorithm, where part (a) is before and part (b) is after the application of the algorithm. The same example solutions of DBSCAN are used. In this case, two solutions are neighboring if they have a degree of similarity of 4. Each set of solutions that have a common identifier is considered a cluster. The best objective function solution inside is considered the center of the cluster. In this example, the solutions *s1*, *s3* and *s4* are considered neighbors. After applying the LP, they were considered as a cluster with the identifier 1 in common. Solution *s1* was chosen as the center. Solution *s2* was an outlier because it did not have neighbors and it was discarded.

Fig. 8 shows an example of the application of the NGI algorithm with solutions of the other two examples. In part (a) there is a minimum spanning tree. In part (b), after the application of the algorithm, it is formed a cluster with solutions *s1*, *s3* and *s4*. The center of the cluster is established as the solution that gets the higher objective function within it. Thus, solution *s1* was chosen as the center of the cluster. Solution *s2* was also seen as outlier and was discarded.

After the application of a clustering method, the local search is applied on the clusters centers. In CS, this application is done only if the cluster has a minimal amount of solutions defined a priori.

The Variable Neighborhood Search method (VND) [37] is the local search component of CS and DCS. The method explores the search space through neighborhood systematic exchange, only accepting solutions when there is an improvement and returning to the first structure when an improvement is detected. In this case, VND was implemented with the heuristics defined in a pre-order execution. VND is used to intensify the search in a particular region considered promising, providing different views of the search space established by the different neighborhood structures.

Three heuristics are used on VND:

- 2-swap First heuristic;
- Labels exchange heuristic;
- Labels exchange heuristic according to the conflicts formed;

The 2-swap First is similar to structure 2 of SA, consisting of traverse vector positions and making an exchange between two labels located at different vector positions. The vector is traversed from position $i = 1$ to position N (N is the vector size). For every position i, the vector is traversed from $j=i+1$ to N and a swap between labels of positions i and j is done.

The labels exchange heuristic is similar to structure 1 of SA, consisting of traverse vector positions from position $i = 1$ to N and each position i and changing the current label for the one that best optimizes the objective function.

The labels exchange heuristic according to the conflicts formed is similar to structure 3 of SA, consisting of traverse vector positions from position $i = 1$ to N and changing the label for the label that produces fewer conflicts with the other positions. Every position that has conflicts with the first position also has its labels changed for those that produce fewer conflicts.

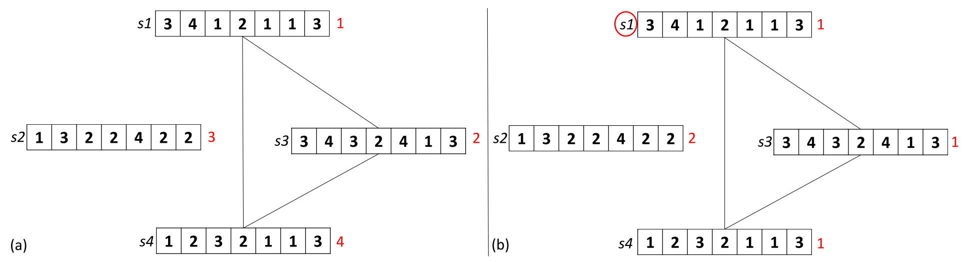


Fig. 7. Example of application of LP algorithm.

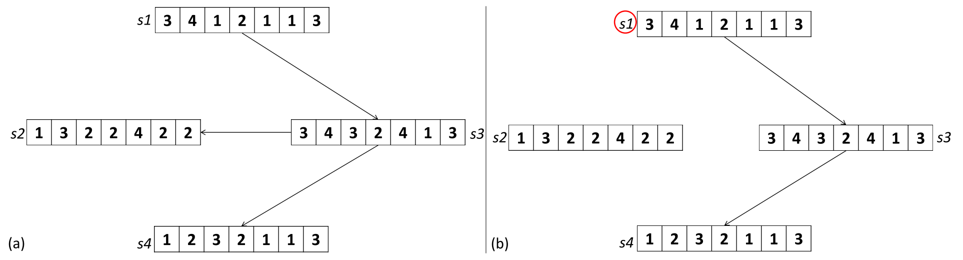


Fig. 8. Example of application of NGI algorithm.

6.2. BRKGA

In the literature, no metaheuristic test results were found for the tested instances in this paper. Thus, tests were performed with the BRKGA metaheuristic [9] to compare the results of CS and DCS.

BRKGA is a heuristic method that has reported good results on different optimization problems. BRKGA is formed by individuals whose genes are real numbers randomly generated within the range]0,1] and are called random-keys.

The decoder function returns a solution to an optimization problem and its respective objective function value from a vector of random keys. The decoder needs to be specific to each optimization problem in which the BRKGA is applied.

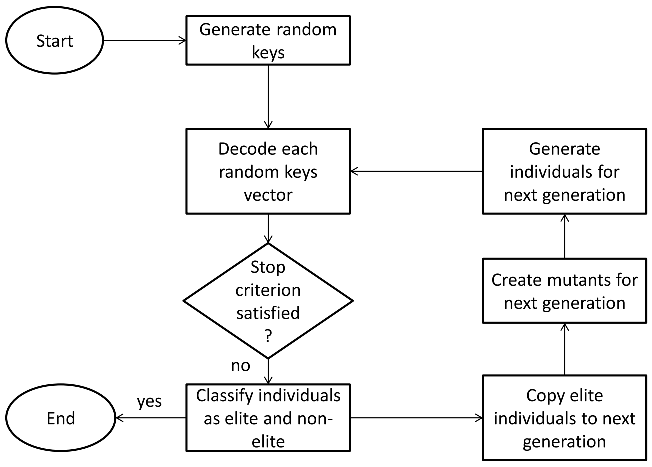
BRKGA starts from a population of random-key vectors that evolve during a number of generations. The initial population consists of v random keys vectors. Each gene of an initial individual is randomly generated within the real interval]0, 1]. Then, the decoder is applied on each individual of the generation g calculating the objective function value. Thus, the population is divided into two groups: an elite group with v_e individuals with the best objective function values, and another group with the remaining $v - v_e$ individuals.

A new generation of individuals must be made to evolve the population. Thus, all elite individuals in generation g are copied to population in the generation $g + 1$.

In the BRKGA a mutant is a vector of random keys generated in the same way as an element of the initial population. A small number (v_m) of mutants is introduced into the new population in each generation. Crossover is also performed by combining pairs of individuals from the current population to produce $v - v_e - v_m$ individuals. In the selection method two parents are randomly chosen. Since one parent necessarily belongs to the set of elite individuals and the other is chosen from the whole population. The parameterized uniform crossover [38] is performed to generate a new offspring. The parameter $\rho_e > 0.5$ privileges that offspring inherit genes from elite parents.

Fig. 9 shows the basic structure of BRKGA.

BRKGA requires that only its decoder be implemented. The rest of code is available in the literature [39]. In this case, it is necessary to design a decoder for the PFCLP problem. Fig. 10 show an example of the proposed decoder function.



Start
Generate random keys
Decode each random keys vector
Generate individuals for next generation
Stop criterion satisfied ?
Create mutants for next generation
no
yes
End
Classify individuals as elite and non-elite
Copy elite individuals to next generation

Fig. 9. BRKGA components.
Source: Adapted from [9].

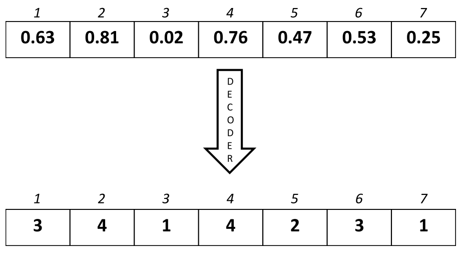


Fig. 10. Example of decoder function for PFCLP.

Each position of the vector represents one point. The value determined at each position of the vector indicates the candidate position (label) chosen. If the value is in the range]0, 0.25] the candidate position 1 is chosen to be labeled. If the value is in the range]0.25, 0.5] the candidate position 2 is chosen. If the value is in the range]0.5, 0.75],the candidate position 3 is chosen. And if the value is in the range]0.75, 1] the candidate position 4 is chosen.

Table 1
Parameters and values used in methods.

Method	Parameter	Description	Interval	Value
SA	*SAmax*	Number of iterations in a Temperature (T)	[500, 1500]	1000
	α	Rate of Temperature decrease	[0.90, 0.99]	0.98
CS	ω_j	Number of solutions in a promising cluster	[5, 30]	15
	NC	Number of clusters	[5, 20]	10
BRKGA	ν	Population size	[100, 1000]	1000
	ν_e	Elite population percentage	[0.10, 0.25]	0.20
	ν_m	Mutant population percentage	[0.05, 0.20]	0.10
	ρ_e	Probability of inheriting key from elite parent	[0.65, 0.80]	0.70

7. Computational results

The tests were performed on instances generated by the method of [21] and we used sets of 505, 5056 and 13 188 points. These data were obtained from a set of points corresponding to cities in the Switzerland map found in http://mistic.heigvd.ch/taillard/problemes.dir/~problemes.html.

There are 12 instances available with labels of different heights and widths in each set (url with available instances will be add after the reviews). A total of 36 instances are tested. All tests were run on a computer with Intel Core i7 3.4 GHz processor and 16 GB memory.

The results in Tables 2, 3, 5 and 7 show the best solutions found in terms of free labels (column RLS, in percentage) and in terms of the minimum distance between labels conflicts (column DMS). We also show the average computational time to find the best solution (column T, in seconds).

Tables 4, 6 and 8 present the average solutions found by the methods in terms of free labels and minimum distance. Calculations are reported in these tables observing the deviation from the best solution (column σ). The deviation was calculated as follows:

$$\sigma = 100\% * \frac{(AverageSolution - BestSolution)}{BestSolution} \tag{7}$$

Figs. 11–13 show the results in box plot diagrams. The rectangles represent the middle half of the solutions for each method, between the lower quartile (25%) and the upper quartile (75%). The lines go from the minimum to the maximum solutions, and the middle line represents the median solution.

7.1. Tuning parameter

The parameters of the CS, DCS and BRKGA methods were calibrated by the Calibra [40]. Calibra is trained through a subset of instances to determine the better parameters for these methods applied to PFCLP. Calibra was run for 200 iterations. Table 1 presents the interval of each parameter recommended by the literature to SA [41] and BRKGA [42] and the value returned by Calibra for each parameter that are used in computational tests.

7.2. Model results

Table 2 presents the best solutions found by the MFBCP [29], MD1 [21], MD2 [21] and MD3 models. All results of the models are update in this article using the CPLEX solver version 12.6. The results of MD2 are different to original article due to a new correct algorithm to counting free labels.

The comparisons of results prioritize the number of free labels, using the minimum distances (dispersion) as a tiebreaker. Thus, in each result that the methods have the same percentage of free labels, the minimum distance is used as a tiebreaker. The results in bold presented in the tables are the best solutions found.

In instances with 505 points it is observed that the new formulation MD3 presents the best results in terms of free labels in eight of 12 tested instances, and in five there was a tie with the other dispersion models. However, in terms of minimum distance, the MD1 model presents the best solutions in all instances. Regarding computational time, the MFBCP formulation had the shortest time to find its solutions. From the instance h04_l16, the MD3 model was limited to 5500 s.

In instances with 5046 points, the MFBCP model presented better results than all other models. Regarding the minimum distance, the MD1 model presented the best results in ten instances and the MD3 in two instances. On the computational time, the MD1 and MD2 models presented the best times in four instances and MD3 in one instance. From the instance h03_l16, the MD3 model was also limited to 5500 s.

7.3. Heuristics results

In this subsection are show the results obtained by the three versions of DCS (with LP, NGI and DBSCAN), CS and BRKGA. All methods were coded in C++and the computational tests carried out on an Intel Core i7 3.4 GHz processor with 16 GB of RAM.

In Table 3 we observed that DCS with LP presented the best solutions in three instances, with NGI in three instances and with DBSCAN in one instance (these results take into account that the minimum distance is used as a tiebreaker on the values of the free labels). CS showed the best results in eight instances and BRKGA in one instance. In the first instance, all methods found solutions with 100% free labels.

In Table 4 we observed that DCS with LP presented the best average solutions in two instances, and with NGI in five instances. DBSCAN and BRKGA did not find the best average for any instance. CS had the best results in two instances. In the first three instances, all methods found the best solutions with 100% free labels except for DBSCAN that did not found in the instance h03_l16. BRKGA had worse average solutions than the other methods.

Thus, the application of CS and DCS with LP and NGI are shown with similar efficiency in finding good solutions for this problem, especially when looking at the results of the box plot diagram (Fig. 11). As it can be seen, the methods are robust in the calculation. However, DCS with DBSCAN presents a higher discrepancy than other DCS versions.

The total computational time increases according to the size of the instance and the number of potential conflicts. CS had the better convergence in most instances with 505 points. This result is due to the fact that Path-relinking is applied to each solution sent to clustering. Because this method is already an application of local search, it finds solutions more quickly than the other methods.

In Table 5 note that DCS with LP presented the best solutions in all instances and with NGI in one instance. DBSCAN, BRKGA and CS did not provide the best solution in any instance. Comparing instances of 505 points, there is an increased complexity of problems contexts and DCS with these two groups already begins to show better results and a good difference in relation to CS. However, the DCS results with DBSCAN are lower than the ones of CS. DCS presented better and different results than the MD3 formulation in terms of free labels in ten instances. These results show that the MD3 formulation needs to be adapt to ensures the optimal number of free labels.

In Table 6 we observed that DCS with LP presented the best average solutions in 12 instances. In addition to obtaining the best solutions, this method is more robust than other methods for application in instances of 5046 points. The observation of the box plot diagram (Fig. 12) proves this statement. As can be seen, DCS

Table 2
Results of computational tests: mathematical formulations.

Instance	MFBCP			MD1			MD2			MD3		
	RLS (%)	DMS (cm)	Time	RLS (%)	DMS (cm)	Time	RLS (%)	DMS (cm)	Time	RLS (%)	DMS (cm)	Time
Instances with 505 points												
h02_l24	100.00	–	0.09	100.00	–	0.98	100.00	–	0.12	100.00	–	0.30
h02_l32	100.00	–	0.28	100.00	–	2.72	100.00	–	1.31	100.00	–	0.19
h03_l16	100.00	–	0.14	100.00	–	2.58	100.00	–	1.17	100.00	–	0.20
h02_l42	99.20	10.04	1.59	99.20	**33.01**	6.06	96.03	1.41	3.85	**99.21**	18.03	0.26
h02_l48	99.20	30.01	2.02	99.20	**39.01**	4.83	92.47	3.16	4.29	**99.21**	39.01	0.22
h03_l24	98.40	11.01	2.41	93.80	**18.11**	19.78	90.09	3.60	2.29	**98.42**	14.04	12.75
h04_l16	96.40	6.08	15.97	85.15	**11.40**	174.98	85.14	2.23	5.79	**96.63**	6.08	5500.00
h03_l28	**97.20**	3.16	4.28	88.30	**19.10**	66.75	87.52	1.41	4.41	97.03	18.03	2790.89
h04_l18	95.20	3.16	6.92	77.23	**12.16**	438.22	81.58	2.23	4.04	**95.25**	0.00	5500.00
h03_l32	**95.60**	3.16	6.01	85.70	**22.10**	41.59	76.43	2.23	34.62	95.45	13.15	5500.00
h04_l21	**93.40**	2.23	17.87	79.21	**14.03**	227.48	–	–	–	92.87	10.20	5500.00
h04_l24	**90.50**	1.41	193.86	64.55	**14.31**	227.75	–	–	–	89.70	10.05	5500.00
Average	*97.09*	*7.81*	*20.95*	*89.36*	*20.36*	*101.14*	*90.93*	*2.32*	*6.19*	*96.98*	*14.29*	*2525.40*
Instances with 5046 points												
h02_l24	**99.64**	1.41	9.23	93.20	**17.02**	10.37	97.68	2.23	10.37	99.45	10.05	4.45
h02_l32	**99.24**	2.23	27.30	90.90	17.02	20.59	95.10	2.23	20.59	99.11	**17.03**	13.80
h03_l16	**98.31**	1.41	52.82	87.80	**10.04**	32.09	93.22	1.41	32.09	98.10	7.28	5500.00
h02_l42	**98.65**	2.23	32.71	83.70	17.02	28.21	90.11	1.41	28.21	98.43	**18.03**	5500.00
h02_l48	**98.07**	1.41	68.08	77.50	**17.02**	58.32	88.82	2.23	58.32	97.90	13.04	5500.00
h03_l24	**95.65**	1.41	5500.00	70.10	**13.03**	5500.00	83.51	1.41	5500.00	95.14	6.08	5500.00
h04_l16	**96.11**	1.41	5500.00	75.40	**6.32**	5500.00	80.89	1.41	5500.00	93.48	2.83	5500.00
h03_l28	**94.26**	1.41	5500.00	75.20	**13.15**	5500.00	77.32	1.41	5500.00	93.48	9.06	5500.00
h04_l18	**92.35**	1.41	5500.00	67.70	**6.32**	5500.00	77.74	1.41	5500.00	91.10	3.16	5500.00
h03_l32	**91.89**	1.41	5500.00	57.40	**12.16**	5500.00	73.26	1.41	5500.00	91.08	11.05	5500.00
h04_l21	**89.73**	1.41	5500.00	49.90	**6.32**	5500.00	–	–	–	88.33	4.12	5500.00
h04_l24	**87.08**	1.41	5500.00	43.20	**5.09**	5500.00	–	–	–	84.44	3.61	5500.00
Average	*95.08*	*1.55*	*3224.18*	*72.67*	*11.71*	*3220.80*	*85.77*	*1.66*	*2764.96*	*94.17*	*8.78*	*4584.85*

Table 3
Results of computational tests (instances with 505 points): best solutions and computational time (in seconds) obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)
h02_l24	**100.00**	–	0.05	100.00	–	0.58	**100.00**	–	0.11	**100.00**	8.06	1.77	**100.00**	–	0.05
h02_l32	**100.00**	–	1.20	100.00	–	1.20	**100.00**	–	0.11	97.82	12.04	1.85	**100.00**	–	0.13
h03_l16	**100.00**	–	1.25	**100.00**	–	1.25	**100.00**	7.28	1.99	94.65	4.47	1.93	**100.00**	–	0.19
h02_l42	**99.21**	3.16	1.38	**99.21**	21.02	1.38	**99.21**	**28.02**	2.24	94.85	13.03	2.01	**99.21**	**28.02**	0.25
h02_l48	**99.21**	34.01	1.44	**99.21**	36.01	1.44	98.81	30.02	2.43	94.25	15.03	2.04	**99.21**	**37.01**	0.24
h03_l24	**98.42**	4.12	2.16	**98.42**	3.16	2.16	97.43	3.16	3.56	90.29	5.09	2.29	**98.42**	**5.10**	0.49
h04_l16	**97.03**	2.24	4.38	**97.03**	**2.83**	3.22	93.86	2.24	5.30	87.12	4.12	2.37	**97.03**	2.24	2.12
h03_l28	**97.23**	**2.24**	6.52	**97.23**	**2.24**	3.59	96.83	3.16	7.55	85.34	6.08	2.52	**97.23**	**2.24**	3.58
h04_l18	**96.04**	1.41	7.03	**96.04**	1.41	4.56	95.05	2.24	9.00	81.78	3.60	2.49	**96.04**	**2.24**	4.98
h03_l32	**95.84**	**2.24**	7.33	**95.84**	1.41	5.80	92.48	3.16	7.53	80.79	6.32	2.71	**95.84**	**2.24**	4.64
h04_l21	**94.65**	**2.24**	8.13	**94.65**	**2.24**	5.76	87.92	2.24	8.93	75.84	4.47	2.71	**94.65**	**2.24**	7.62
h04_l24	**92.08**	**2.24**	9.28	**92.08**	1.41	7.19	86.73	2.24	9.91	69.70	4.47	2.93	92.08	**3.16**	8.77
Average	*97.48*	*5.99*	*4.18*	*97.48*	*7.97*	*3.18*	*95.69*	*8.37*	*4.89*	*87.70*	*7.23*	*2.30*	*97.48*	*9.39*	*2.75*

Table 4
Results of computational tests (instances with 505 points): average solutions and deviation obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)
Instances with 505 points															
h02_l24	**100.00**	–	0.00	100.00	–	0.00	**100.00**	–	0.00	**100.00**	–	0.00	**100.00**	–	0.00
h02_l32	**100.00**	–	0.00	100.00	–	0.00	**100.00**	–	0.00	97.82	12.04	0.00	**100.00**	–	0.00
h03_l16	**100.00**	–	0.00	**100.00**	–	0.00	99.12	4.54	0.88	94.65	4.47	0.00	**100.00**	–	0.00
h02_l42	**99.21**	3.16	0.00	**99.21**	4.95	0.00	98.41	8.56	0.81	94.85	3.07	0.00	**99.21**	**13.23**	0.00
h02_l48	**99.21**	16.34	0.00	**99.21**	**21.73**	0.00	98.36	9.05	0.86	94.25	9.07	0.00	99.13	20.58	0.08
h03_l24	**98.42**	1.69	0.00	**98.42**	**1.75**	0.00	95.87	1.75	2.61	90.29	2.82	0.00	98.39	2.50	0.03
h04_l16	**97.03**	2.24	0.00	**97.03**	**2.30**	0.00	93.13	1.62	4.16	87.12	3.34	0.00	**97.03**	2.24	0.00
h03_l28	97.19	1.74	0.04	97.19	2.24	0.04	96.44	1.99	0.82	85.31	6.08	0.04	**97.21**	1.87	0.02
h04_l18	95.98	1.41	0.06	**96.00**	1.41	0.04	94.05	1.66	2.09	81.75	3.60	0.04	95.99	1.46	0.05
h03_l32	**95.74**	1.50	0.10	95.68	1.41	0.17	91.15	1.79	5.07	80.66	6.32	0.17	95.68	1.66	0.17
h04_l21	**94.40**	2.24	0.27	94.38	2.24	0.29	86.91	1.62	8.81	75.62	4.47	0.29	94.30	2.24	0.38
h04_l24	91.86	1.50	0.24	**91.94**	1.41	0.15	83.65	1.46	9.71	69.60	4.47	0.15	91.64	1.50	0.47
Average	*97.42*	*3.53*	*0.06*	*97.42*	*4.38*	*0.06*	*94.76*	*3.40*	*3.58*	*85.41*	*4.77*	*0.07*	*97.38*	*5.25*	*0.10*

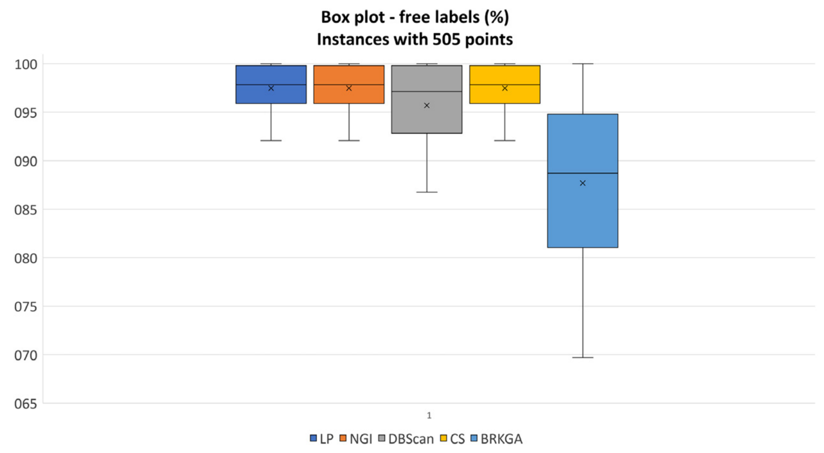


Fig. 11. Box plot diagram: free labels (%) for instances with 505 points.

Table 5
Results of computational tests (instances with 5046 points): best solutions and computational time (in seconds) obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)
Instances with 5046 points															
h02_l24	**99.64**	9.06	11.58	99.64	9.06	15.09	97.94	1.41	16.99	73.00	1.41	17.70	87.87	1.41	20.00
h02_l32	**99.27**	6.08	14.61	99.17	11.05	10.47	94.93	1.41	18.46	65.22	1.41	18.50	83.87	1.41	20.00
h03_l16	**98.39**	1.41	23.79	98.36	1.41	18.09	92.94	1.41	21.52	63.15	1.41	19.30	94.33	1.41	30.00
h02_l42	**98.73**	3.16	18.94	98.65	2.24	27.65	91.30	1.41	26.50	58.20	1.41	20.10	92.15	1.41	30.00
h02_l48	**98.12**	3.16	22.66	98.04	3.16	25.39	89.54	1.41	36.29	54.65	1.41	20.40	90.43	1.41	30.00
h03_l24	**96.16**	1.41	34.77	95.86	1.41	29.95	85.77	1.41	42.41	49.82	1.41	22.90	87.53	1.41	40.96
h04_l16	**95.12**	1.41	38.44	94.89	1.41	40.95	84.03	1.41	39.75	49.46	1.41	23.70	86.92	1.41	45.00
h03_l28	**95.05**	1.41	41.95	94.81	1.41	52.50	81.87	1.41	46.93	44.47	1.41	25.20	85.12	1.41	50.00
h04_l18	**93.62**	1.41	45.25	93.36	1.41	47.21	80.90	1.41	55.19	45.02	1.41	24.90	83.99	1.41	60.00
h03_l32	**93.06**	1.41	49.22	92.85	1.41	54.64	78.16	1.41	60.80	40.80	1.41	27.10	82.40	1.41	56.00
h04_l21	**91.68**	1.41	56.17	91.48	1.41	61.26	76.71	1.41	62.98	40.13	1.41	27.30	80.96	1.41	61.00
h04_l24	**89.36**	1.41	66.32	89.04	1.41	73.61	72.35	1.41	72.70	36.18	1.41	29.30	77.09	1.41	71.00
Average	*95.68*	*2.73*	*35.31*	*95.51*	*3.07*	*38.07*	*85.54*	*1.41*	*41.71*	*51.68*	*1.41*	*2.72*	*86.05*	*1.41*	*42.83*

Table 6
Results of computational tests (instances with 5046 points): average solutions and deviation obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)
Instances with 5046 points															
h02_l24	**99.64**	3.73	0.00	98.57	2.67	1.07	97.04	1.41	2.66	72.22	1.41	1.07	87.26	1.41	14.09
h02_l32	**99.22**	2.67	0.05	96.62	2.02	2.67	94.31	1.41	5.23	63.55	1.41	2.57	82.85	1.41	19.57
h03_l16	**98.33**	1.41	0.07	95.57	1.41	2.88	92.51	1.41	6.33	61.36	1.41	2.83	86.91	1.41	12.18
h02_l42	**98.65**	1.80	0.08	97.02	1.58	1.73	90.82	1.41	8.66	57.24	1.41	1.65	81.52	1.41	18.68
h02_l48	**98.01**	1.88	0.11	95.80	1.71	2.37	88.68	1.41	10.55	53.40	1.41	2.29	79.26	1.41	20.86
h03_l24	**96.09**	1.41	0.07	93.05	1.41	3.24	85.18	1.41	12.80	48.36	1.41	2.93	81.60	1.41	16.62
h04_l16	**95.00**	1.41	0.13	93.43	1.41	1.79	83.77	1.41	13.52	48.70	1.41	1.54	81.48	1.41	15.70
h03_l28	**94.87**	1.41	0.19	94.50	1.41	0.58	81.16	1.41	16.96	44.32	1.41	0.33	79.05	1.41	18.79
h04_l18	**93.51**	1.41	0.11	92.98	1.41	0.68	80.59	1.41	16.11	44.84	1.41	0.41	78.10	1.41	18.48
h03_l32	**92.95**	1.41	0.13	92.46	1.41	0.65	77.31	1.41	20.16	40.63	1.41	0.42	76.02	1.41	20.69
h04_l21	**91.49**	1.41	0.21	91.00	1.41	0.74	76.22	1.41	20.15	39.92	1.41	0.53	76.15	1.41	19.18
h04_l24	**89.14**	1.41	0.24	88.75	1.41	0.68	71.68	1.41	24.43	36.06	1.41	0.32	72.32	1.41	22.10
Average	*95.57*	*1.78*	*0.11*	*94.15*	*1.61*	*1.59*	*84.94*	*1.41*	*13.13*	*13.13*	*1.41*	*1.41*	*80.21*	*1.41*	*18.08*

with LP and NGI algorithms are robust in the calculation. However, DCS with DBSCAN, CS and BRKGA presented a high discrepancy.

DCS with LP converges faster to the best solutions in nine instances and NGI in three instances. Thus, in addition to presenting the best results DCS with LP found good solutions faster. In the context of problems with 5046 points for the PFCLP, it performs better than the other methods, especially in comparison with CS.

We can note that the advantage in solving the PFCLP by DCS is that the MD3 formulation and formulations of [21] and [29] cannot reach and ensure optimal solutions to the problem in reasonable time. Also, due to memory limitations, the formulations failed to achieve optimal results. Thus, the use of metaheuristics was required and, as noted, DCS obtained the best results. Because of the limitations presented in the use of the formulations, the instances with 13 188 points were only tested with metaheuristics.

In Table 7, DCS with LP presented the best solutions in five instances, and with NGI in seven instances. These two approaches also differed from CS, which presented solutions close to DCS with

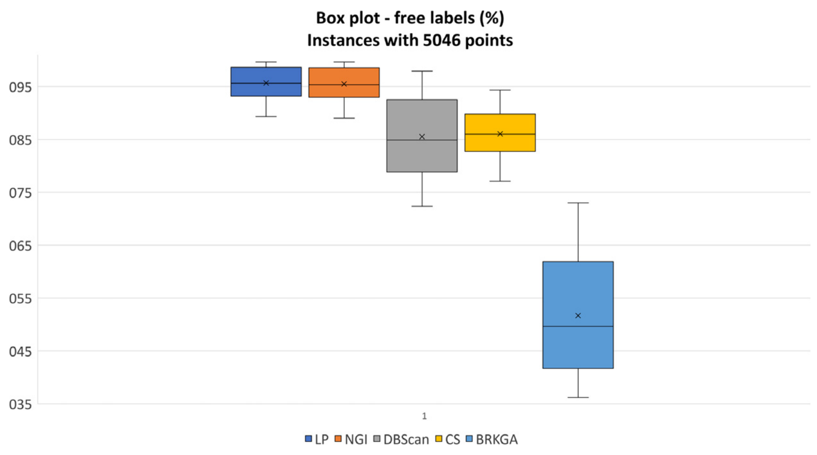


Fig. 12. Box plot diagram: free labels (%) for instances with 5046 points.

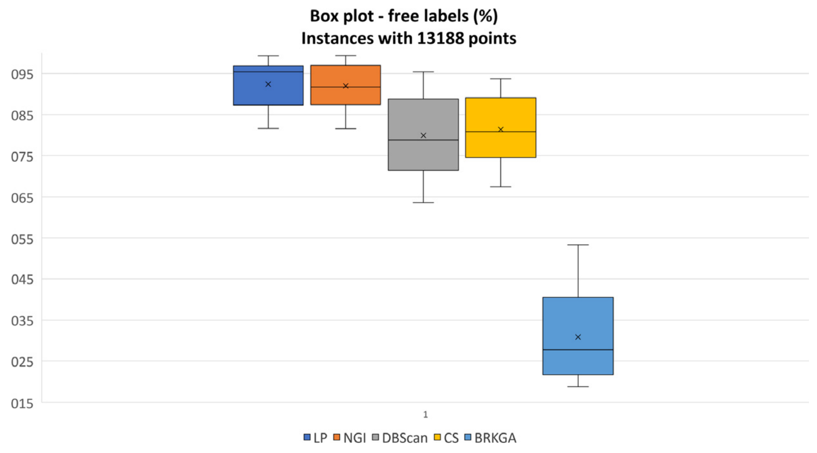


Fig. 13. Box plot diagram: free labels (%) for instances with 13 188 points.

DBSCAN. BRKGA had the worst results. In Table 8, it was found that DCS with LP presented the best average solutions in seven instances and with NGI in five instances. The other methods did not show best average solutions in any instance.

These tables show that the DCS solutions with LP and NGI were best and different from the CS results. However, tests with DBSCAN are shown worse. As in the 5046 points test, DCS with LP showed the best results. But in this context, the tests with NGI also stood out with relevant results. Thus, DCS with NGI is also efficient to find good solutions to PFCLP. The box plot diagram (Fig. 13) confirms the same conclusion.

As it can be seen, the methods LP and NGI are robust in the calculation. We can noted that DCS with LP converges rapidly to the best solutions in nine instances and NGI in three instances. In this sense, DCS with LP can get results faster than other methods in the studied instances. we can observe that DCS can overcome the CS in these results. Instances of 5046 and 13 188 points are more complex and with larger labels, hence DCS show to have a better performance than CS to return good results with a better computational time.

7.4. *The Wilcoxon statistical test*

The Wilcoxon statistical test (WSR, Wilcoxon signed-rank test) [43] was performed to analyze the performance of the methods. This test is used to compare sets of solutions and to investigate whether there is significant differences between the solutions found by DCS and CS. Table 9 shows the results reported by the Wilcoxon statistical test. WSR indicates that the DCS solutions, with the exception of DBSCAN, were significantly better than the CS solutions according to the presented *V* and *p-value* values. This result suggests that the use of DCS on the set of instances gives better results compared with CS when LP and NGI are used.

[33] also applied the CS to solve PFCLP. However, our application differs from its when working with the distance between labels. Another difference is the instances used by [33] that do not have this distance information.

8. Conclusion

This paper presented and tested the hybrid method CS, which presented good results in different contexts involving combinatorial optimization. The method aims to find promising areas of the search space to intensify the search in these regions to find good solutions. The IC component of CS requires the use of a clustering method to identify promising regions. The quality of the method is important to detect these regions efficiently.

We observed that the IC component does not have theoretical and practical basis. Therefore, this paper has proposed a solution to these issues, finding clustering algorithms that form clusters with the necessary degree of quality to apply the local search. We tested these methods on the PFCLP problem, seeking to prove that this adaptation is efficient.

The DBSCAN, LP and NGI algorithms were tested in adaptation with the clustering process of CS. This adaptation aims to detect promising regions by the density caused by the clustering of solutions, so it was called DCS.

Table 7
Results of computational tests (instances with 13 188 points): best solutions and computational time (in seconds) obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)	RLS (%)	DMS (cm)	T (s)
Instances with 13 188 points															
h02_l24	99.32	1.41	496.44	**99.33**	1.41	309.04	95.38	1.41	579.32	53.29	1.41	517.73	93.73	1.41	580.91
h02_l32	98.21	1.41	549.86	**98.24**	2.24	478.39	92.13	1.41	599.47	43.48	1.41	541.13	91.82	1.41	601.19
h03_l16	96.94	1.41	582.01	**96.97**	1.41	612.50	89.16	1.41	699.66	42.13	1.41	564.63	89.84	1.41	703.09
h02_l42	96.64	1.41	647.11	**96.97**	1.41	652.78	87.59	1.41	699.48	35.71	1.41	587.93	86.87	1.41	701.01
h02_l48	95.43	1.41	685.81	**96.97**	1.41	656.15	85.42	1.41	859.60	31.69	1.41	596.70	86.28	1.41	863.20
h03_l24	**95.43**	1.41	698.17	92.39	1.41	813.85	79.98	1.41	949.59	27.49	1.41	669.83	81.93	1.41	951.11
h04_l16	**95.43**	1.41	705.78	90.99	1.41	773.66	77.66	1.41	949.59	28.08	1.41	693.23	79.68	1.41	952.20
h03_l28	**89.79**	1.41	763.56	89.72	1.41	833.11	75.11	1.41	949.81	23.17	1.41	737.70	76.99	1.41	955.56
h04_l18	88.46	1.41	757.36	**88.54**	1.41	849.77	73.57	1.41	949.68	24.78	1.41	728.33	75.74	1.41	953.08
h03_l32	86.94	1.41	809.04	**87.02**	1.41	935.57	70.72	1.41	1219.87	20.60	1.41	792.68	74.16	1.41	1221.01
h04_l21	**85.05**	1.41	821.11	84.94	1.41	960.46	68.73	1.41	1219.93	21.23	1.41	794.86	71.88	1.41	1221.01
h04_l24	**81.61**	1.41	872.33	81.59	1.41	1037.03	63.57	1.41	1219.86	18.78	1.41	857.03	67.43	1.41	1218.36
Average	*92.44*	*1.41*	*699.05*	*91.97*	*1.48*	*742.69*	*79.92*	*1.41*	*907.99*	*30.87*	*1.41*	*63.50*	*81.36*	*1.41*	*910.14*

Table 8
Results of computational tests (Instances with 13 188 points): average solutions and deviation obtained by DCS, CS and BRKGA methods.

Instance	DCS									BRKGA			CS		
	LP			NGI			DB								
	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)	RLS (%)	DMS (cm)	Dev. (%)
Instances with 13 188 points															
h02_l24	**99.30**	1.41	0.04	99.28	1.41	0.05	95.02	1.41	4.52	52.15	1.41	1.14	88.60	1.41	11.45
h02_l32	98.15	1.41	0.10	**98.19**	1.47	0.05	91.71	1.41	7.09	41.28	1.41	2.20	84.29	1.41	15.20
h03_l16	96.86	1.41	0.11	**96.87**	1.41	0.10	88.82	1.41	9.14	39.78	1.41	2.36	85.34	1.41	12.94
h02_l42	96.52	1.41	0.47	**96.71**	1.41	0.27	87.17	1.41	11.19	34.54	1.41	1.17	78.06	1.41	21.76
h02_l48	**95.35**	1.41	1.70	96.09	1.41	0.91	84.47	1.41	14.63	30.26	1.41	1.43	86.01	1.41	12.70
h03_l24	**93.87**	1.41	1.63	92.30	1.41	3.38	79.29	1.41	20.17	25.90	1.41	1.59	81.29	1.41	17.25
h04_l16	**93.11**	1.41	2.42	90.87	1.41	5.01	77.14	1.41	23.55	27.22	1.41	0.86	79.34	1.41	20.18
h03_l28	**89.63**	1.41	0.18	89.62	1.41	0.19	74.48	1.41	20.38	23.03	1.41	0.14	71.97	1.41	23.16
h04_l18	88.35	1.41	0.22	**88.36**	1.41	0.21	73.30	1.41	20.72	24.59	1.41	0.19	67.27	1.41	28.09
h03_l32	86.85	1.41	0.20	**86.88**	1.41	0.15	70.25	1.41	23.71	20.47	1.41	0.14	73.35	1.41	18.43
h04_l21	**84.90**	1.41	0.19	84.83	1.41	0.26	68.28	1.41	24.41	21.07	1.41	0.16	71.31	1.41	19.13
h04_l24	**81.48**	1.41	0.16	81.44	1.41	0.21	63.18	1.41	29.00	18.69	1.41	0.09	66.26	1.41	22.76
Average	*92.03*	*1.41*	*0.62*	*91.79*	*1.42*	*0.90*	*79.43*	*1.41*	*17.38*	*17.38*	*1.41*	*0.96*	*77.76*	*1.41*	*18.59*

Table 9
Wilcoxon test.

	505			5046			13 188		
	LP	NGI	DBSCAN	LP	NGI	DBSCAN	LP	NGI	DBSCAN
V	1233	1198	0	28 920	28 910	25 615	28 920	28 920	16 061
p-value	2.20E−16	2.20E−16	1	2.20E−16	2.20E−16	2.20E−16	2.20E−16	2.20E−16	2.20E−16

Sets of instances with 505, 5046 and 13 188 points to label in a graphic device were selected for testing. DCS was proven to be as effective as CS to find good solutions in the test with 505 points and found similar results. In instances with 5046 and 13 188 points DCS proved to be better than CS in the search of good solutions and computational time. The LP and the NGI algorithms excelled with the best results, especially on the larger set of instances with 13 188 points.

LP and NGI algorithms improved the identification of promising regions, which provided better results than those presented by CS.

In forthcoming work, DCS should be applied to other optimization problems and LP and NGI algorithms should be operated together with other metaheuristics, for example, with a population metaheuristic.

Acknowledgments

Funding: This work was supported by the São Paulo Research Foundation (FAPESP) [grant number 2014/00580-0 and 2016/07135-7] and the National Counsel of Technological and Scientific Development (CNPQ) [grant number 482170/2013-1 and 301836/2014-0].

References

[1] I.H. Osman, J.P. Kelly, Meta-Heuristics: Theory and Applications, Springer Science & Business Media, 2012.
[2] A.C.M.d. Oliveira, A.A. Chaves, L.A.N. Lorena, Clustering search, Pesqui. Oper. 33 (1) (2013) 105–121.
[3] A.A. Chaves, L.A.N. Lorena, Clustering search algorithm for the capacitated centered clustering problem, Comput. Oper. Res. 37 (3) (2010) 552–558.
[4] A.A. Chaves, F. de Assis Correa, L.A.N. Lorena, Clustering search heuristic for the capacitated p-median problem, in: Innovations in Hybrid Intelligent Systems, Springer, 2007, pp. 136–143.
[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231.
[6] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007) 036106.
[7] J.F.R. Viana, M.J.N. Gomes, A.F.S. Xavier, A New Hierarchical Procedure for Natural Groups Identification in Euclidean Space, Tech. rep., Federal university of Ceará, 2012.
[8] J. Christensen, J. Marks, S. Shieber, An empirical study of algorithms for point-feature label placement, ACM Trans. Graph. 14 (3) (1995) 203–232.
[9] J.F. Gonçalves, M.G. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (5) (2011) 487–525.

[10] A.C. Oliveira, L.A. Lorena, Detecting promising areas by evolutionary clustering search, in: Brazilian Symposium on Artificial Intelligence, Springer, 2004, pp. 385–394.
[11] A.C. Oliveira, L.A. Lorena, Hybrid evolutionary algorithms and clustering search, in: Hybrid Evolutionary Algorithms, Springer, 2007, pp. 77–99.
[12] G. Ribeiro Filho, M.S. Nagano, L.A.N. Lorena, Evolutionary clustering search for flowtime minimization in permutation flow shop, in: International Workshop on Hybrid Metaheuristics, Springer, 2007, pp. 69–81.
[13] R.M. de Oliveira, G.R. Mauri, L.A.N. Lorena, Clustering search for the berth allocation problem, Expert Syst. Appl. 39 (5) (2012) 5499–5505.
[14] E.J. Araújo, A.A. Chaves, L.L. de Salles Neto, A.T. de Azevedo, Pareto clustering search applied for 3d container ship loading plan problem, Expert Syst. Appl. 44 (2016) 50–57.
[15] F. Glover, R. Martí, Fundamentals of scatter search and path relinking, Control Cybernet. 39 (2000) 653–684.
[16] S. Liu, Z.-T. Dou, F. Li, Y.-L. Huang, A new ant colony clustering algorithm based on DBSCAN, in: Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, Vol. 3, IEEE, 2004, pp. 1491–1496.
[17] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Tech. rep., School of Computer Science, Carnegie-Mellon University, 2002.
[18] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (10) (2010) 103018.
[19] A. Devillez, P. Billaudel, G.V. Lecolier, A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition, Fuzzy Sets and Systems 128 (3) (2002) 323–338.
[20] E.J. Araujo, Hybrid Method with Detection of Promising Areas Based on Density for Point Feature Cartographic Label Placement (Master's thesis), Universidade federal de Sao Paulo, 2016.
[21] S.P. Gomes, G.M. Ribeiro, L.A.N. Lorena, Dispersion for the point-feature cartographic label placement problem, Expert Syst. Appl. 40 (15) (2013) 5878–5883.
[22] G.M. Ribeiro, L.A.N. Lorena, Heuristics for cartographic label placement problems, Comput. Geosci. 32 (6) (2006) 739–748.
[23] S. Zoraster, The solution of large 0–1 integer programming problems encountered in automated cartography, Oper. Res. 38 (5) (1990) 752–759.
[24] T. Strijk, B. Verweij, K. Aardal, et al., Algorithms for Maximum Independent Set Applied to Map Labelling, Tech. rep., Department of Computer Science, Utrecht University, 2000.
[25] P.K. Agarwal, M. Van Kreveld, S. Suri, Label placement by maximum independent set in rectangles, Comput. Geom. 11 (3) (1998) 209–218.
[26] M. Yamamoto, G. Camara, L.A.N. Lorena, Tabu search heuristic for point-feature cartographic label placement, GeoInformatica 6 (1) (2002) 77–90.
[27] M. Yamamoto, L.A. Lorena, A constructive genetic approach to point-feature cartographic label placement, in: Metaheuristics: Progress as Real Problem Solvers, Springer, 2005, pp. 287–302.
[28] G.L. Cravo, G.M. Ribeiro, L.A.N. Lorena, A greedy randomized adaptive search procedure for the point-feature cartographic label placement, Comput. Geosci. 34 (4) (2008) 373–386.
[29] G.M. Ribeiro, L.A.N. Lorena, Lagrangean relaxation with clusters for point-feature cartographic label placement problems, Comput. Oper. Res. 35 (7) (2008) 2129–2140.
[30] L. Li, H. Zhang, H. Zhu, X. Kuai, W. Hu, A labeling model based on the region of movability for point-feature label placement, ISPRS Int. J. Geo-Inf. 5 (9) (2016) 159.
[31] A. Marín, M. Pelegrín, Towards unambiguous map labeling-Integer programming approach and heuristic algorithm, Expert Syst. Appl. 98 (2018) 221–241.
[32] J.-H. Haunert, A. Wolff, Beyond maximum independent set: an extended integer programming formulation for point labeling, ISPRS Int. J. Geo-Inf. 6 (11) (2017) 342.
[33] R.L. Rabello, G.R. Mauri, G.M. Ribeiro, L.A.N. Lorena, A clustering search metaheuristic for the point-feature cartographic label placement problem, European J. Oper. Res. 234 (3) (2014) 802–808.
[34] G.R. Mauri, G.M. Ribeiro, L.A. Lorena, A new mathematical model and a Lagrangean decomposition for the point-feature cartographic label placement problem, Comput. Oper. Res. 37 (12) (2010) 2164–2172.
[35] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[36] A.A. Chaves, L.A.N. Lorena, A preprocessing phase for the evolutionary clustering search, in: Workshop on Computational Intelligence (WCI), 2006, p. 1.
[37] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (11) (1997) 1097–1100.
[38] W.M. Spears, K.A.D. Jong, On the virtues of parameterized uniform crossover, in: Proc. of the Fourth International Conference on Genetic Algorithms, 1991, pp. 230–236.
[39] R.F. Toso, M.G. Resende, A c++ application programming interface for biased random-key genetic algorithms, Optim. Methods Softw. 30 (1) (2015) 81–93.
[40] B. Adenso-Diaz, M. Laguna, Fine-tuning of algorithms using fractional experimental designs and local search, Oper. Res. 54 (1) (2006) 99–114.
[41] B. Suman, P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization, J. Oper. Res. Soc. 57 (10) (2006) 1143–1160.
[42] H. Prasetyo, G. Fauza, Y. Amer, S.H. Lee, Survey on applications of biased-random key genetic algorithms for solving optimization problems, in: Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on, IEEE, 2015, pp. 863–870.
[43] M. Neuhäuser, Wilcoxon–mann–whitney test, in: International Encyclopedia of Statistical Science, Springer, 2011, pp. 1656–1658.