

Universitat de Girona

Escola Politècnica Superior



MASTER THESIS

DECISION SUPPORT METHODS FOR GLOBAL OPTIMIZATION

BY

FERRAN TORRENT-FONTBONA

ADVISORS

VÍCTOR MUÑOZ I SOLÀ

BEATRIZ LÓPEZ IBÁÑEZ

2012

Abstract

Immobile location-allocation (LA) problems is a type of LA problem that consists in determining the service each facility should offer in order to optimize some criterion (like the global demand), given the positions of the facilities and the customers. Due to the complexity of the problem, i.e. it is a k^n combinatorial problem (where k is the number of possible services and n the number of facilities) with a non-convex search space with several sub-optimums, traditional methods cannot be applied directly to optimize this problem. Thus we proposed the use of clustering analysis to convert the initial problem into several smaller sub-problems. By this way, we presented and analyzed the suitability of some clustering methods to partition the commented LA problem. Then we explored the use of some metaheuristic techniques such as genetic algorithms, simulated annealing or cuckoo search in order to solve the sub-problems after the clustering analysis.

Acknowledgements

I would like to thank all people that helped me during this Master's Thesis, specially to both of my supervisors Víctor and Bea and all people from Newronia and eXiT (Control Engineering and Intelligent Systems) research group.

I also would like to give thanks to my parents and my girlfriend for their emotional support.

Table of Contents

Abstract	iii
Acknowledgements.....	v
Table of Contents	vii
List of Tables.....	xi
List of Figures	xiii
Acronyms and abbreviations.....	xv
CHAPTER 1. Introduction.....	17
1.1. Motivation and problem statement.....	17
1.1.1. Data	18
1.1.2. Contributions.....	18
1.2. Outline.....	19
CHAPTER 2. State of the art	21
2.1. Clustering	21
2.1.1. K-means.....	22
2.1.2. Hierarchical clustering.....	24
2.1.3. Region growing.....	26
2.1.4. Fuzzy C-Means.....	27
2.1.5. Possibilistic C-Means	27
2.1.6. Competitive learning.....	28
2.1.7. Genetic algorithm based clustering	28

Decision support methods for global optimization

2.1.8.	Density-based clustering.....	29
2.1.9.	Multi-way clustering.....	29
2.1.10.	Affinity propagation	30
2.1.11.	Quality indices for clustering evaluation.....	30
2.1.12.	Summary clustering.....	33
2.2.	Location-allocation problem	33
2.2.1.	Classes	34
2.2.2.	Models.....	35
2.2.3.	Summary Location-allocation.....	37
2.3.	Optimization.....	38
2.3.1.	Complete optimization methods	39
2.3.2.	Incomplete optimization methods.....	44
2.3.3.	Summary optimization	50
2.4.	Optimization methods applied to location-allocation problem.....	52
2.5.	Summary	53
CHAPTER 3.	Clustering	55
3.1.	K-means.....	56
3.2.	Lloyd's algorithm	59
3.3.	Region Growing	61
3.4.	Agglomerative hierarchical clustering.....	64
3.5.	Genetic algorithm based clustering	66
3.6.	Affinity propagation	69
3.7.	Discussion.....	70
CHAPTER 4.	Location-allocation.....	75
4.1.	Mathematical model	76
4.2.	Genetic algorithm.....	78

Decision support methods for global optimization

4.3. Simulated annealing.....	80
4.4. Cuckoo search	85
4.5. Comparison	87
CHAPTER 5. Conclusions.....	91
CHAPTER 6. Future work.....	93
Appendix A. Numeric distances.....	95
Appendix B. Cluster images	97
Bibliography	107

List of Tables

Table 2.1. Main clustering techniques.	33
Table 2.2. Some examples of research about location-allocation.	37
Table 2.3. Optimization methods.	51
Table 2.4. Summary of Location-Allocation latest research.	52
Table 3.1.K-means results	57
Table 3.2. Lloyd's algorithm results.....	59
Table 3.3. Region growing results	62
Table 3.4. Hierarchical clustering results	65
Table 3.5. Genetic clustering results	68
Table 3.6. Affinity propagation results.....	70
Table 3.7. Clustering algorithms results achieved from a dataset of 15578 Catalan bars.....	72
Table 4.1. Fitness of the final solution found by GA using different populatin sizes.....	79
Table 4.2. Elapsed time (s) by GA with different population sizes.....	80
Table 4.3. LA results using SA with different neighborhood functions. Best results are in bold face.....	84
Table 4.4. Fitness of the different solutions found by CS using different number of nests	86
Table 4.5. LA results. Best results are in bold face.....	88
Table 4.6. LA results using SA initialized with individual LA solution and SA initialized randomly	89

List of Figures

Figure 1.1. Bars' geographical positions.....	19
Figure 2.1. K-means algorithm process.....	23
Figure 2.2. Hierarchy example in hierarchical clustering.....	25
Figure 2.3. Agglomerative hierarchical clustering.....	25
Figure 2.4. Divisive hierarchical clustering.....	25
Figure 2.5. Region growing example. On the left, original image. On the right, segmented image.....	26
Figure 2.6. Clustering of geographical points using region growing.....	27
Figure 2.7. Neural Network example.....	28
Figure 2.8. Illustration of how affinity propagation works. Taken from [42].....	30
Figure 2.9. Optimization methods classification.....	38
Figure 2.10. Depth-First Search example. Continuous arrows represent the direction of the search and discontinuous arrows represent the backtracks.....	40
Figure 2.11. Breath-First Search example.....	41
Figure 2.12. Graph example to illustrate A* mechanism.....	43
Figure 3.1. Clustering result using Algorithm 3.1. The number of clusters in the dense regions is higher.....	58
Figure 3.2. Clustering result using Algorithm 3.2.....	58
Figure 3.3. Calinski index vs. number of clusters using Algorithm 3.2.....	59
Figure 3.4. Clustering result using Algorithm 3.3.....	60
Figure 3.5. Calinski index vs. number of clusters using Lloyd's algorithm.....	61
Figure 3.6. Clustering result using Region Growing and $D_{max} = 1\text{km}$	62

Figure 3.7. Clustering result using Region Growing and $D_{max} = 2\text{km}$	62
Figure 3.8. Clustering result using Region Growing and $D_{max} = 5\text{km}$	63
Figure 3.9. Hierarchical clustering results using different Th_{jump}	64
Figure 3.10. Clustering result using hierarchical clustering.	65
Figure 3.11. Chromosome example. The $K_i = 5$ centroids are distributed along the $L = 10$ slots	66
Figure 3.12. Single point crossover	66
Figure 3.13. Genetic clustering	67
Figure 3.14. Fitness evolution (vertical axis) of the best individual vs. generation (horizontal axis)	69
Figure 3.15. Clustering result using Affinity propagation	69
Figure 4.1. Example of LA.	75
Figure 4.2. Probability function of changing a match given the occupation of the bar.....	81

Acronyms and abbreviations

ACO: Ant Colony Optimization

B&B: Branch & Bound

BFS: Breath-First Search

CS: Cuckoo Search

CSA: Clonal Selection Algorithm

DCSP: Distributed Constrain Satisfaction Problem

DFS: Depth-First Search

FA: Firefly Algorithm

FCM: Fuzzy C-Means

GA: Genetic Algorithms

GBS: Gradient Based Search

HC: Hill Climbing

LP: Linear Programming

PCM: Possibilistic C-Means

PSO: Particle Swarm Optimization

SA: Simulated Annealing

SO: Spiral Optimization

TS: Tabu Search

CHAPTER 1. Introduction

1.1. Motivation and problem statement

Nowadays a vast number of matches of different sports are played in the same week, and some of them are played simultaneously. All that matches are also broadcast by different TV channels due to the current great interest on sport. Traditionally a lot of people go to the closest bar to watch the desired match. Nevertheless, sometimes different people go to the same bar to see different matches at the same time, what is a decision problem for the barman. This problem has taken relevance due to the great mobility of the people around the world and an evidence of that is the development of tools like *Wewatchthematch* to know which bars around you broadcast your desired match. As an example, the said application *Wewatchthematch* has more than 1 million users and several signed bars from most cities around the world, like Barcelona, Washington, New York, Amsterdam, Milano, Roma, etc.

The challenge is to build a system able to decide the best match for each signed bar depending on the *Wewatchthematch* users. These users are supposed to tell which is their desired match and their current location using the GPS receiver incorporated in their smartphone. For taking into account the users that forget to tell their desired match and those people that do not have the application, it is needed an estimator of the demand for every match depending on historic data. Once the demand is known, or accurately estimated, it is time to work out the match each bar should broadcast in order to maximize the number of people that goes to the bars to watch the sport events, i.e. in order to maximize the global demand.

In this Master's thesis we try to give an answer to the barman decision problem making an automatic system able to find the best match for every bar depending on the people around every bar. For example, let's suppose that in the city A there are 10 bars and there also are 300 people that want to watch the match X and 100 that want to watch the match Y, and both matches are played at the same time. Probably, the barmen would choose the match X because it has more demand. Thus, if we suppose that customers are uniformly distributed among the bars, each bar would have 30 customers. However, if eight bars broadcast match X and 2 match Y, they would have 37.5 and 50 customers respectively, what improves the occupation of all bars.

This decision making problem is a type of location-allocation problem. LA is a decision making problem that consists in finding the optimal situation of some facilities over the bounded

space and finding, at the same time, the best allocation of the customers in the facilities minimizing the distance, the cost or/and the time. In this case, the facilities are the bars and the aim is to find the type of each facility or the service each facility offers in order to maximize the global amount of customers and minimize the distance between the customers and the facilities.

Note that we are dealing with a great amount of data (bars and customers) what makes unfeasible to solve such problem only using optimization algorithms. Indeed it is needed to simplify the initial problem or to partition it. Thus we also explore the applicability of clustering techniques to partition the LA problem in order to simplify the optimization algorithms work.

1.1.1. Data

The data used correspond to 15578 bar's positions from Catalunya taken from *Páginas Amarillas*. This dataset is only a subset of the real dataset that has bars and customers from around the World. This fact increases the need of partition it. The bars addresses have been converted into geographical coordinates using *googleMaps* (see Figure 1.1). The demand is generated with a random simulation where for each bar a random number of customers between 0 and 30 is generated. Then each customer decides the match it wants to watch according a certain probability function.

1.1.2. Contributions

As contributions, this work provides a state of the art of clustering techniques, LA problem modeling and optimization techniques, classifying them according their completeness and their type of search.

Moreover, as LA is a complex problem and unfeasible for amount of facilities we have to deal with, we have considered that the best option is to segment the problem in several smaller problems performing a clustering of the facilities according to their positions. Thus, we contribute with the use of clustering techniques in order to simplify a given LA problem and providing an analysis of the applicability of some clustering methods for this purpose.

Finally, we contribute giving an analysis and comparison of how GA, SA and CS solve the stated problem, defining a new neighborhood function for SA and CS that does not need a coordinate system.

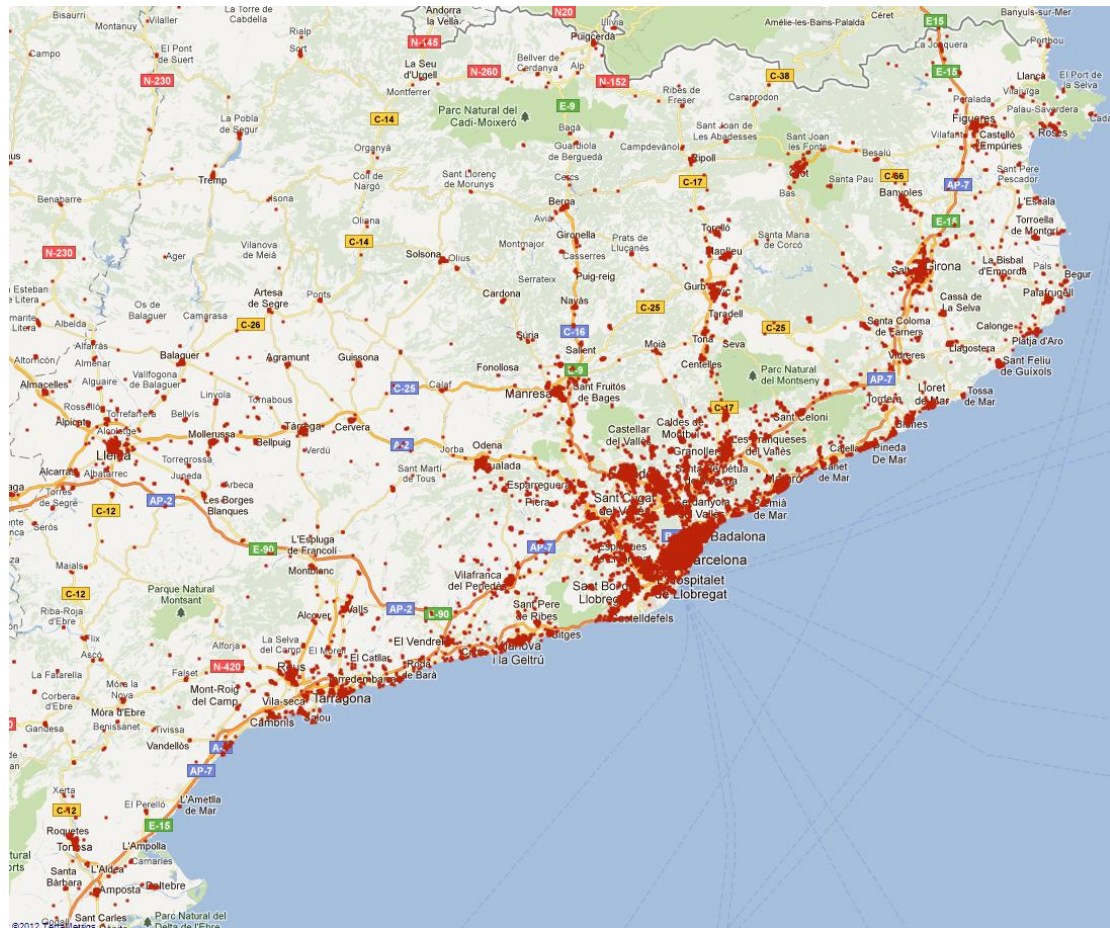


Figure 1.1. Bars' geographical positions

1.2. Outline

First of all, this Master's thesis presents a state of the art of the clustering methods, making a qualitative comparison between the different presented methods. Then it expounds the different types of location-allocation problem and a state of the art of their mathematical modeling. Finally, in Chapter 2, it explains the different optimization methods making a qualitative comparison between them and presenting a brief state of the art of the optimization methods.

In Chapter 3, it is presented the implementation of some clustering methods and the results obtained from them. We used the results obtained to make a quantitative comparison of the different clustering methods.

Chapter 4 exposes three optimization techniques, GA, SA and CS and its applicability to solve the LA problem. Then, a quantitative comparison of their performances in different cases is made.

Finally, Chapter 5 provides the conclusions reached in this Master's Thesis and Chapter 6 proposes some future work.

CHAPTER 2. State of the art

This chapter surveys relevant research done in the areas of clustering, location-allocation modeling and optimization. These three areas are the most relevant for our work in order to be applied to the problem studied in this thesis. Location-allocation modeling has great importance since it defines the problem and determines the variables and its weight and relationship that will mathematically model the location-allocation problem that, afterwards, is going to be solved using an optimization technique. Clustering is used to partition the whole dataset into smaller parts to transform the global location-allocation optimization problem into several smaller and simpler location-allocation problems.

First of all, main clustering techniques are explained with a brief state of the art. Then, different location-allocation models are presented and some variations of these models that some researchers have done in last years. Finally, main optimization techniques are pointed out, classifying them into two groups: complete optimization techniques and incomplete optimization techniques.

2.1. Clustering

The beginning of cluster analysis is fuzzy, but it can be said that Sokal and Sneath incited a world-wide research on clustering methods with their monography 'Principles of numerical taxonomy' [1] in 1963. In their work they developed, for the first time, the concept of numerical taxonomy which deals with the classification of taxonomic units¹ based on patterns of overall similarities or based on branching the patterns of their estimated evolutionary history.

The proofs that 'Principles of numerical taxonomy' incited a world-wide research on clustering are later publications of books such as 'Les bases de la classification automatique' [2] in 1970, 'Mathematical taxonomy' [3] in 1971, 'Cluster analysis for applications' [4] in 1973, 'Cluster analysis' [5] in 1973, 'Automatische Klassifikation' [6] in 1974, 'Empirische Verfahren zur Klassifikation' [7] in 1974, 'Probleme und Verfahren der numerischen Klassifikation' [8] in 1975, 'Cluster-Analyse-Algorithmen' [9] in 1975 and 'Clustering algorithms' [10] in 1975, or articles

¹ Groups of one or more organisms that share some features.

such as 'The detection of disease clustering and a generalized regression approach' [11] in 1967, 'Hierarchical clustering schemes' [12] in 1967 and 'A new approach to clustering' [13] in 1969. This clustering fever had the consequence that the basic problems and methods of clustering and its possible applications became well-known in the scientific community.

Clustering is a fundamental data analysis method. It is an unsupervised method which seeks to partition a set of objects into groups (clusters), so the objects within a cluster are similar but different from the rest of objects of the other groups. Thus, the clusters must be homogeneous but different from each other. Clustering is used in many fields like:

- Biology and bioinformatics: to describe and to make spatial and temporal comparisons of communities of organisms, [14] and [15].
- Medicine: to differentiate different types of tissue and fluid or different regions in a 3D image, [16].
- Business and marketing: in market research, [17], or for grouping shopping items.
- Sociology: for social network analysis, [18].
- Climatology: to find weather regimes or atmospheric patterns, [19].
- Geographical clustering.

Clustering can be achieved by several algorithms, that differ in their notion of what constitutes a cluster and how to find the clusters. Thus, there are many possible classifications of these algorithms depending on the features we focus. However, if this feature is the relationship between clusters, there are two possible categories for clustering:

- Hard clustering, where each item cannot belong to two or more clusters.
- Soft clustering or fuzzy clustering, where each item belongs to each cluster to a certain degree, hence an item can be assigned to several clusters partially.

In hard clustering, the most popular methods are k-means, [20], [21] and [22], and hierarchical clustering, [23], [24].

The most popular methods in fuzzy clustering are the fuzzy c-means proposed by Bezdek [25] and the possibilistic c-means proposed by Krishnapuram and Keller [26] and [27]. Both methods are inspired by k-means, but they introduce some techniques to perform the fuzzy clustering.

2.1.1. K-means

The standard k-means algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse code modulation, not as a clustering technique, and it was not published until 1982 [28]. Despite the age of the algorithm, it is one of the most popular clustering techniques in the present day. K-means aims to divide the dataset into a given number k of clusters so as to minimize the within-cluster squares sum.

$$\arg \min_S \left\{ \sum_{i=1}^k \sum_{x_j \in S_i} d^2(x_j - \mu_i) \right\} \quad (2.1)$$

where x_j is the j th item, S_i is the i th cluster, μ_i the mean of its items and S is the set of clusters. Figure 2.1 shows how k-means algorithm works.

Algorithm 2.1. K-means algorithm

- (1) K (3 in Figure 2.1) points (circles) are set arbitrarily. These points represent the cluster centroids.
 - (2) Objects (squares) are assigned to the closest point making a cluster (red, green, blue)
 - (3) Centroids are recalculated using the members of each cluster.
 - (4) Objects are reassigned to the closest cluster centroid.
 - (5) Steps 3 and 4 are repeated until convergence has been reached or until a certain number of iterations have passed.
-

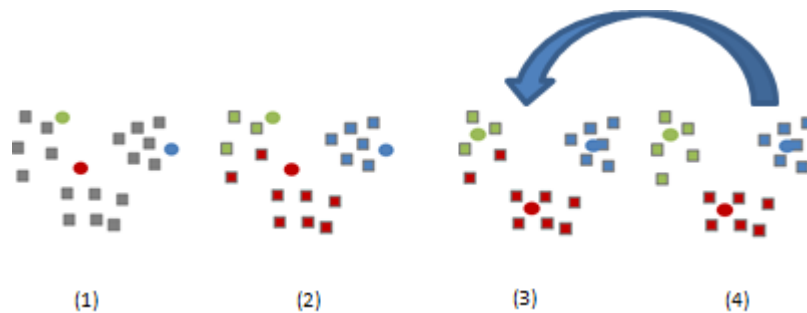


Figure 2.1. K-means algorithm process

K-means algorithm has inspired other clustering techniques such as Lloyd's algorithm or k-means++ [29].

Lloyd's algorithm, as k-means, was developed for Stuart Lloyd. Actually they are the same algorithm (sometimes both algorithms are referred as Lloyd's algorithm) with the difference that k-means initially plants k seeds randomly across the data space and Lloyd's algorithm starts by partitioning the input objects into k initial sets. Then, as in k-means, the centroids of each group are calculated and objects are reassigned to its closest cluster (step 4 in Algorithm 2.1).

K-means++ (see Algorithm 2.2) also focus on solving the problem of initialization; in fact, like Lloyd's algorithm it only differs from traditional k-means on the initialization.

Algorithm 2.2. K-means++ algorithm

1. Initialization
 - a. Choose one center uniformly at random from among the objects.
 - b. For each object x , compute $D(x)$, the distance between x and the nearest center that has already been chosen.
 - c. Choose a new object at random as a new center, according a weighted probability distribution where a point x is chosen with probability proportional to $D^2(x)$.
 - d. Repeat steps 1.2 and 1.3 until k centers have been chosen.
 2. Objects are assigned to the closest point making a cluster.
 3. Centroids are recalculated using the members of each cluster
 4. Objects are reassigned to the closest cluster centroid
 5. Steps (3) and (4) are repeated until convergence has been reached or until a certain number of iterations have passed.
-

2.1.2. Hierarchical clustering

Hierarchical clustering seeks to work out the clustering building a hierarchy of clustering results (see Figure 2.2), where each level differs on the maximum distance between clusters. Once the hierarchy is made, the algorithm chooses the best result according a criterion. The level to divide the dataset is chosen depending on the distance between levels. There are two different types of strategies to build the hierarchy:

- *Agglomerative*: each item has its own cluster and algorithm merge pairs of clusters as it goes up through the hierarchy. To decide the appropriate hierarchy level and so, the adequate number of clusters, there is not a universal criterion. However most researchers use a criterion based on the agglomerative distance in agglomerative clustering as stated Mojena [30]. Thus, it consists of finding the first big enough jump between two hierarchy levels so, the joint items or clusters are different enough to not group them together. Mojena mathematically stated it as following:

$$\frac{\alpha_g - \bar{\alpha}}{\sigma_\alpha} \geq K, K = \{2.5, 3.5\} \quad (2.2)$$

Where α_g is the agglomerative distance (distance between two items to consider they are equal), $\bar{\alpha}$ the mean of the agglomerative distances and σ_α its standard deviation. The constant K depends on the researcher criterion but the most popular values are 2.5 and 3.5. Figure 2.3 illustrates the agglomerative hierarchical clustering process.

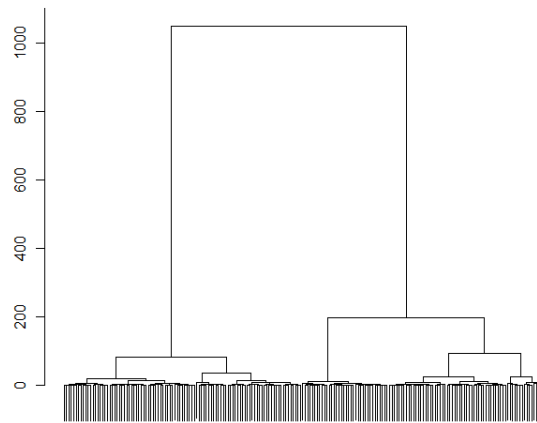


Figure 2.2. Hierarchy example in hierarchical clustering



Figure 2.3. Agglomerative hierarchical clustering

- Divisive*: all items start in the same cluster and then the algorithm performs splits as it goes down through the hierarchy. To decide the appropriate number of clusters in divisive hierarchical clustering one can use Mojena’s criteria as in agglomerative hierarchical clustering. However, it should be taken into account that the best hierarchy level is not the one which presents a big enough jump but the last one as the hierarchy is tracked in the opposite way by the algorithm. The process is illustrated in Figure 2.4.

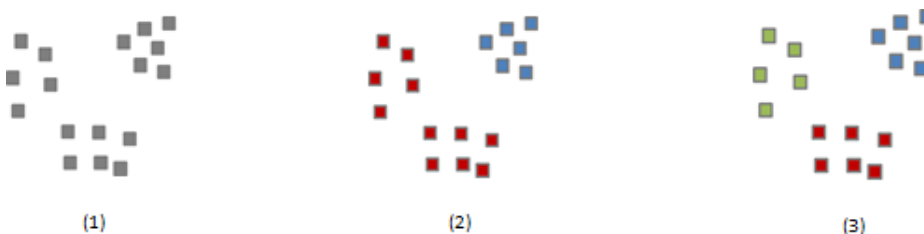


Figure 2.4. Divisive hierarchical clustering

2.1.3. Region growing

Region growing is a segmentation method. It is widely used in image processing. It consists of planting different seeds across the data space. Each seed is the starting point of the region (cluster) that expands itself incorporating the neighboring elements similar to the elements that belong to the region. As k-means, this algorithm is highly depending on the initial conditions, i.e. for different starting points (seeds) the algorithm will find different results.

However, region growing can be done sequentially expanding one cluster until the differences between objects is greater than a threshold, then expanding the next cluster and so on until all objects are assigned to a cluster. This implementation needs a threshold to determine when to stop expanding a region but it does not need the number of clusters. Also, it always finds the same result. In this document we will refer to such region growing.

This algorithm is totally different from the others presented in this chapter as it does not find groups of elements; it finds groups of connected elements that share a similar level of energy or intensity. For example, Figure 2.5 shows how Region growing interprets as the same region the upper half of the lightning due to the pixels in this part share the same intensity (white). It also shows that the elements of the same region must be connected. Note the algorithm divides the lightning in two parts due to the cloud that hides part of the lightning.



Figure 2.5. Region growing example. On the left, original image. On the right, segmented image.

Clustering different points in the data space using region growing (see Figure 2.6) means that the algorithm will put in the same cluster all connected elements. The connected elements are those that have a neighbor closer than a certain distance threshold. Therefore, the empty space between elements works as frontier and divides the dataset in different clusters.

The main problem of this algorithm is its lack of robustness against outliers as a few isolated elements may connect two different clusters.

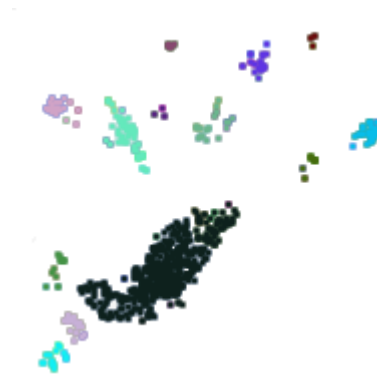


Figure 2.6. Clustering of geographical points using region growing.

2.1.4. Fuzzy C-Means

FCM follows the same process of k-means algorithm showed in Figure 2.1. However, in FCM, any point x has a set of coefficients giving the degree of being in the k th cluster $w_k(x)$. Hence, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster. Thus, centroid of each cluster is computed using the whole dataset instead of only its member as k-mean does, as follows:

$$c_k = \frac{\sum_x w_k(x) x}{\sum_x w_k(x)} \quad (2.3)$$

The degree of belonging, $w_k(x)$, is related inversely to the distance from x to the cluster center as calculated on the previous pass. It can also depend on another parameter used to control the weight given to the closest center.

2.1.5. Possibilistic C-Means

PCM is a FCM variation. PCM algorithm tends to identify meaningful cluster as defined by dense regions rather than partition the whole data set into k parts. It overcomes the need to specify the number of clusters and it is robust in presence of noise and outliers as they do not form dense regions. However, it requires a good initialization that can be done by FCM. Thus, PCM is a refinement of FCM algorithm. This refinement can be explained for the PCM tending to find dense regions and because in PCM the memberships can be interpreted as degrees of typicality¹ instead of degrees of sharing as in FCM. Thus, two items of a given cluster that are equidistant from the centroid of the cluster can be significantly different and two items in a given cluster can be equal even though the two points are arbitrarily far away from each other. Thereby the possibilistic approach simply means that the membership value of a point in a cluster represents the typicality of the point in the class, or the possibility of the

¹ Typicality describes the amount of properties of a cluster that an object has.

point belonging to the class. Hence, these fuzzy approaches are based on k-means idea, but they compute the cluster centroids using the information of all dataset, instead of each cluster items.

Since these algorithms were proposed, several researchers have applied them to their research fields: bioinformatics (microarray data clustering) [31] and [32], for genes classification based on the measurements of their expression levels, social networks [33], etc. Others have extended these algorithms to different data type, such as non-spatial data as in [34].

2.1.6. Competitive learning

Another interesting clustering method is competitive learning [35]. Competitive learning is a type of unsupervised learning in artificial neural networks, in which nodes compete for the right to respond to a subset of the input data, namely, each node of the network tries to respond only to one subset (cluster) of the whole dataset. Thus, competitive clustering seeks to specialize the nodes of neuronal network in order to induce neurons to respond to a subset of the data, so once the network has been trained it would be able to partition the input data (see Figure 2.7).

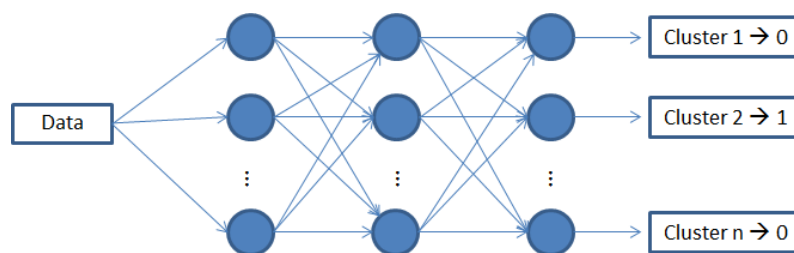


Figure 2.7. Neural Network example

SOM (Self Organizing Maps) or SOFM (Self Organizing Feature Maps) are an example of competitive learning as they are a type of artificial neural networks trained using competitive learning. SOM are able to produce a low-dimensional clustering of a dataset.

2.1.7. Genetic algorithm based clustering

Clustering, as it has been stated before, has the objective to partition a dataset without any information about its structure. However, most clustering techniques require certain parameters, such as the number of clusters and the cluster shapes although these parameters are not known in most real-life situations. This problem can be dealt with by employing evolutionary approaches, [36]. The capability of GA, [37], is applied to evolve the proper

number of clusters and to provide appropriate clustering. The advantages of GA clustering bring several researchers implement GA clustering algorithm for their work, [38] and [36].

Initially, GA creates a random set of solutions called population. In clustering each solution is achieved using different user required parameters like the number of clusters, the initial positions of the clusters centroids, etc. Then the GA makes the population to evolve, improving the quality of the solutions, until a sufficient good solution is found or it exceeds a certain amount of time. The GA procedure is deeper explained in subsection 2.3.2.5.

2.1.8. Density-based clustering

Density based clustering is a technique developed first by Ester *et al.* [39] in 1996 in order to provide a clustering technique capable of working on large databases with a minimal domain knowledge to determine the input parameters and capable of discovering clusters with arbitrary shape with good efficiency.

The algorithm detects clusters according to the density of the region, so clusters would be separated from each other by contiguous regions of low density of elements. Elements located in these low-density regions are typically considered noise or outliers. This property makes this technique robust against the presence of noise.

Density based clustering may provide the same result as region growing is there are not outliers in the dataset, i.e. the frontiers between regions are empty of objects. It overcomes the problem of outliers in the dataset. However, it considers as outlier all isolated objects removing them from the dataset, what may not be desirable. Density-based clustering has some similarities with PCM in the sense of both identify clusters identifying dense regions, nevertheless, PCM is a fuzzy technique and density-based clustering is a hard clustering technique.

2.1.9. Multi-way clustering

Elements to be clustered are often formed by a combination of related heterogeneous components (features). For example, a document is made of words, titles, authors, citations, etc. Co-clustering aims to cluster the features and the elements of the dataset simultaneously to identify the subset of features where the resulting clusters are meaningful according to a certain evaluation criterion. This problem was first studied by Hartigan in 1972 [40] under the name of direct clustering.

Co-clustering was extended to multi-way clustering [41] to cluster a set of objects by simultaneously clustering their heterogeneous components. Indeed, the problem is much more challenging because different pairs of features may participate in different types of similarity relationships. Additionally, some relations may involve more than two components.

2.1.10. Affinity propagation

A newly clustering technique is to exchange messages between the elements of the dataset in order to find the most representative ones called exemplars and then build the clusters defined by them. This technique was proposed in 2007 by Frey and Dueck [42] to provide an efficient pattern detection algorithm and they called it Affinity propagation.

Affinity propagation technique simultaneously considers all elements as potential exemplars. It views every element as a node in a network that can transmit real-valued messages along edges of the network that tell how good is each element representing another element. Then the most representative elements progressively emerge and the rest of elements are linked to one exemplar. Figure 2.8 shows how this methods works. First of all there are an amount of nodes linked by edges. Then the nodes begin to exchange messages (arrows) and the exemplar begin to emerge (change their color from green to black and then red).

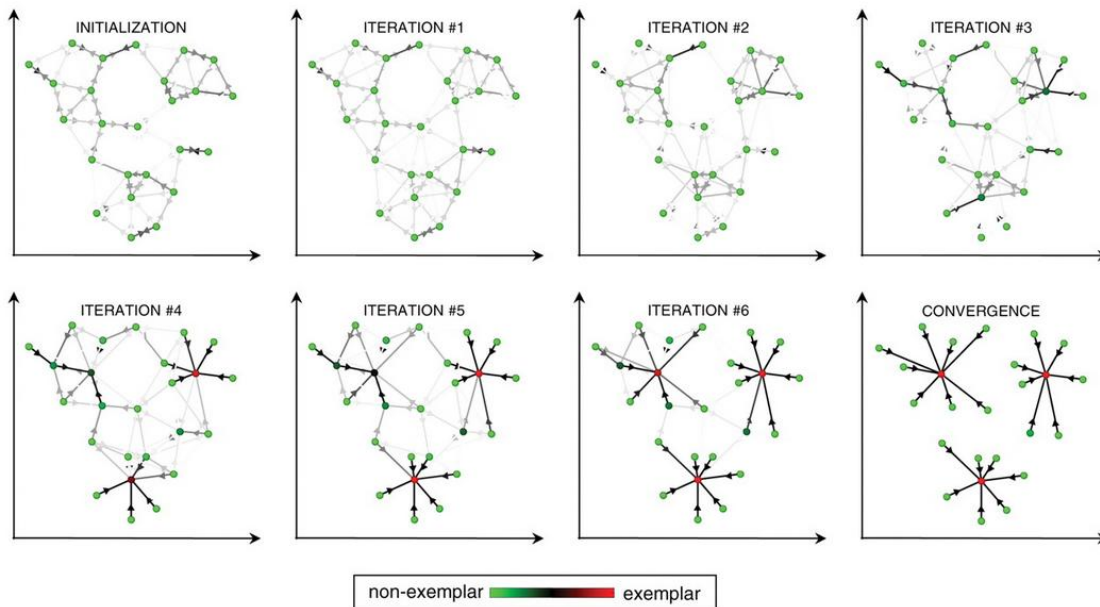


Figure 2.8. Illustration of how affinity propagation works. Taken from [42].

2.1.11. Quality indices for clustering evaluation

To evaluate the clustering quality, there is not a universal index but there are several indices developed by different researchers. One of the most popular is the Calinski index, [43]. It evaluates the homogeneity inside each cluster and the heterogeneity between clusters; the greater $C(k)$ the better.

$$C(k) = \frac{B(k) / k - 1}{W(k) / n - k} \quad (2.4)$$

$$B(k) = \sum_{i=1}^k n_i \|z_i - z\|^2 \quad (2.5)$$

$$W(k) = \sum_{i=1}^k \sum_{j=1}^{n_i} \|x_j - z_i\|^2 \quad (2.6)$$

$$z = \frac{\sum_{i=1}^n x_i}{n} \quad (2.7)$$

Where K is the number of groups, n the number of items, $B(k)$ the sum of square differences between clusters, $W(k)$ the sum of square differences inside the cluster, z is the global centroid of the data, z_i is the i th cluster centroid and x_j is the j th member of the cluster. This index can be used by non-fuzzy and fuzzy clustering.

Hartigan index $H(k)$, [40] and [44], is another index used to evaluate clustering quality. The main weakness of this index is that the number of clusters must be between 1 and 10. Oppositely to Calinski index, the lower $H(k)$ the better and it only uses the differences inside the clusters.

$$H(k) = \frac{\frac{W(k)}{W(k+1)} - 1}{n - k - 1} \quad (2.8)$$

Krzanowski Lai index, [45], is another quality index. As Hartigan it is useful to compare different clusters of the same dataset using different number of clusters. In fact Hartigan and Krzanowkis Lai indices are useful to find the optimum number of cluster of a dataset, not the optimum clustering. It is defined as

$$KL(k) = \left| \frac{DIFF(k)}{DIFF(k+1)} \right| \quad (2.9)$$

Where

$$DIFF(k) = (k-1)^{2/p} \cdot W(k-1) - k^{2/p} \cdot W(k) \quad (2.10)$$

And p is the number of features of the dataset. The optimum k value is the one that maximizes $KL(k)$.

Kaufman and Rousseeuw, [46], presented another clustering quality index $KR(k)$. The optimum k is the one that maximizes $KR(k)$. The index is defined as

$$KR(k) = \frac{1}{n} \sum_{i=1}^n s(i) \quad (2.11)$$

Where $s(i)$ is the 'silhouette' of the i th element and it is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (2.12)$$

Being $a(i)$ the mean of differences between the i th element and the members of its cluster and $b(i)$ the minimum distance between x_i and the other clusters ($k' \neq k$) defined as $b(i, k')$.

$$a(i) = \frac{\sum_{j=1, j \neq i}^{n_k} d(x_i, x_j)}{n_k - 1} \quad (2.13)$$

$$b(i) = \min\{b(i, k')\} \quad (2.14)$$

$$b(i, k) = \frac{\sum_{j=1}^{n_k} d(x_i, x_j)}{n_k} \quad (2.15)$$

In 1979 David L. Davies and Donald W. Bouldin introduced another metric for evaluating clustering algorithms called Davies-Bouldin index, [47]. The aim is to minimize this index $DB(k)$, where k is the number of clusters.

$$DB(k) = \frac{1}{k} \sum_{i=1}^k R_{i,qt} \quad (2.16)$$

$$R_{i,qt} = \max_{j, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{d_{ij,t}} \right\} \quad (2.17)$$

$$d_{ij,t} = \sqrt[t]{\sum_{s=1}^p |z_{is} - z_{js}|^t} = z_i - z_j \quad (2.18)$$

$$S_{i,q} = \sqrt[q]{\frac{1}{n_i} \sum_{j=1}^{n_i} x_j - z_{i2}^q} \quad (2.19)$$

Note that to evaluate clustering quality it must be defined a distance in order to evaluate the similarity or dissimilarity between elements. There is a vast amount of distances, the most important ones are mentioned at 0.

2.1.12. Summary clustering

We reviewed the main clustering techniques mentioning their main strengths and weaknesses and when they were developed (see Table 2.1). We also classified them as hard or soft clustering techniques. Finally we introduced a collection of indices used to measure the quality of a clustering.

Table 2.1. Main clustering techniques.

Algorithm	A priori information	Initial conditions dependence	Type of clustering	Other features
K-means (1957)	Number of clusters	Strong	Hard	
Hierarchical clustering (1973)	None	None	Hard	The appropriate number of clusters is calculated once the hierarchy has been build.
Region growing	None	Moderate	Hard	Weak against outliers.
FCM (1984)	Number of clusters	Strong	Soft	
Lloyd's algorithm (1986)	Number of clusters	Strong	Hard	Approximation of the k-means algorithm.
Genetic algorithm based clustering (1992)	None	Low	Hard and soft	
PCM (1993)	None	Strong	Soft	Based on FCM Robust against noise and outliers
Competitive clustering (1993)	None	None	Hard	Requires a training of the system.
Density-based clustering (1996)	None	Low	Hard	Efficient with large datasets
Multi-way clustering (2005)	None	None	Hard	Useful with heterogeneous data Efficient with large datasets
K-means++ (2006)	Number of cluster	Moderate	Hard	Attenuates the k-means algorithm dependency on initial conditions.
Affinity propagation (2007)	None	Low	Hard	

2.2. Location-allocation problem

In 1909, Alfred Weber, [48], located a single warehouse by minimizing the total travel distance between the warehouse and a set of spatially distributed customers. Thus, he solved a problem known as location problem that consists of estimating the position(s) to locate or supply center(s) to serve the customers. He also started the problem known as location theory. This work was reconsidered by Isard, [49], in 1956, with his study on industrial location, land use and related problems. Finally, in 1963, Leon Cooper, [50], extended the location problem to several facilities, stating the problem known as location-allocation (LA) problem. LA problem consists of, first selecting the locations of a number of facilities to minimize the global distance or another optimization criterion, and then to decide the corresponding allocation of the customers to facilities. LA problem is a common problem in urban infrastructural

constructions, such as hospitals, schools, parks, fire stations, police stations or telecommunication networks, in industrial location and warehouse location or even military purposes such as Openshaw and Steadman, [51], (1982) study where they proposed an optimal nuclear bombing strategy based on population data.

The simplest LA cost function considers just the distance between customers and facilities [52], [53], [54], [55]; therefore, it minimizes the global distance between them. It can be mathematically formulated as

$$\min_X \left\{ \sum_{i=1}^m \sum_{j=1}^n z_{ij} w_i d_{ij} \right\} \quad (2.20)$$

$$\text{s.t. } \sum_{j=1}^n z_{ij} = 1 \quad (2.21)$$

$$z_{ij} \in \{0,1\} \quad (2.22)$$

where d_{ij} is the distance between the customer i and facility j , Z_{ij} is one if the customer i is assigned to the facility j , X are the coordinates of the new facilities and w_i is the weight assigned to customer i . The algorithm has the objective of determining the best coordinates to locate the facilities so that all customers have accessibility to one of the facilities. To include the travel time and cost, it just has to add a function to model time and a function to model the travel cost and their corresponding weights to specify the importance of each term.

2.2.1. Classes

LA techniques differ in the decision variable(s), the type of distance used, the system parameters, etc. However, Church, [56], (1999) classified the LA problems in four general classes:

- **Median:** median, or P-median, models identify the median points among the potential points so that the total weighted distance can be minimized. The mathematical formulation is (2.20). In this class it is mandatory to serve all the demand; however, the facilities are expected to have infinite capacity.
- **Covering:** these models intend to find facilities which provide customers the access to facility service within a specified distance. The facilities want to cover maximum customer to reach their target. These models are used, for example, for wireless tower establishing for network. The algorithms used to solve this problem do not have to provide coverage for all customers.
- **Capacitated:** capacitated models consider the capacity of each facility with respect to the demand. The optimization is reached considering all the demand must be served; therefore, these models try to avoid the problem of demand overflow.
- **Competitive:** competitive models deal with the question of locating facilities to provide a service (or goods) to the customers in a given geographical area where other

facilities offering the same service are already present. Thus, the new facilities will have to compete for the market with the existing facilities.

Out of this classification, there is the immobile version of the problem [57], which consists in locating services when facilities and customers are known. It consists in determining the service each facility offers in order to optimize the demand taking into account the competitors. This Master's Thesis deals with this LA problem.

2.2.2. Models

LA problem is not an old-fashioned problem. As a proof of this, in the last years several researchers have presented their approaches for LA. Most researchers focus on implementing algorithms to speed up the optimization process due to the computational complexity of the optimization process. However, in this section we focus on the LA problems modeling.

In 2004, Hsieh and Tien [58] presented a study of un-capacitated LA problems with rectilinear distances using Kohonen self-organizing feature maps (SOFM). SOFM are a type of neural networks that are trained with unsupervised learning to produce a low dimensional representation of the input data. They use them to do an initialization of the optimum search. The cost function in their study was (2.20) where $d_{ij} = |x_i - a_j| + |y_i - b_j|$ is the rectilinear distance between customer i and facility j . Also in 2004, Li and Yeh [59] proposed a method to solve un-capacitated LA problem using GA and a Geographical Information System (GIS). They used the same formulation of Hsieh and Tien with the difference that they employed Euclidean distances.

In 2005, Kongsomsaksakul *et al.* [60] presented a shelter location-allocation problem for flood evacuation. They posed the shelter location as a Stackelberg game¹. The leader role is played by the authority and it selects the shelter locations to minimize the total evacuating time. They formulated the problem as a bi-level programming where the upper level is the location problem that models the authority's role. This level is formulated as

$$\min_{\mathbf{X}} \left\{ \sum_a v_a(\mathbf{X}) t_a(v_a(\mathbf{X})) \right\} \quad (2.23)$$

$$\text{s.t. } X_j = \begin{cases} 1 & \text{if shelter } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad (2.24)$$

Where $\mathbf{X} = [X_1 \ X_2 \ \dots \ X_N]$ and X_j represents if the shelter is or not selected, v_a is the traffic flow in link a and $t_a(v_a)$ is the travel time in link a which is a function of v_a . In the

¹ Stackelberg game is a strategic game where the leader firm moves first and then the other firms follow the leader.

lower level problem they proposed a combined distribution and assignment model to model the evacuees' decision.

Pang and Feng [61] proposed, in 2006, a competitive LA model and they solved it using the CSA. In their model the target was to maximize the market share of p new facilities located in an area with k competitive facilities. They define the market share, M_j , captured by facility j as

$$M_j = \sum_{i=1}^n b_i \frac{A_j / d_{ij}^\lambda}{\sum_{r=1}^k A_r / d_{ir}^\lambda} \quad (2.25)$$

Where b_i is the buying power of customer i , A_j is the attractive level of facility j and d_{ij} is the distance between customer i and facility r . If it is considered that the incoming franchise that wants to put the p new facilities has c facilities already located in the area, then the target function that models the total market share, $M(X)$, by the incoming franchise is

$$M(X) = \sum_{i=1}^n b_i \frac{\sum_{r=1}^c \frac{A_r}{d_{ir}^\lambda} + \sum_{s=1}^p \frac{A_s^N}{d_i^\lambda(X_s)}}{\sum_{r=1}^k \frac{A_r}{d_{ir}^\lambda} + \sum_{s=1}^p \frac{A_s^N}{d_i^\lambda(X_s)}} \quad (2.26)$$

Where X are the coordinates of the p new facilities, A_s^N is the attractive level of the new s th facility and $d_i(X_s)$ is the distance between customer i and coordinates X_s . Redondo *et al.* also proposed a competitive LA problem method using formulation (2.26) but they used the UEGO algorithm instead of the clonal selection algorithm of Pang and Feng.

Abolian *et al.* [62] presented another interesting research about competitive LA in 2007. They proposed a mathematical model that considers a gravity model, like the before mentioned competitive LA researches, but with elastic concave demand and multiple design characteristics.

Yasenovskiy and Hodgson proposed in 2007 a hierarchical location-allocation modeling for hierarchical health centers location. They add hierarchy levels to the formulation (2.20). Thus, the minimizing function can be written as

$$\min_X \left\{ \sum_{k \in K} \sum_{i=1}^{m_k} \sum_{j=1}^{n_k} U_k z_{ij} w_i d_{ij}^k \right\} \quad (2.27)$$

Where k is the hierarchy level, U_k is the proportion of total demand at level k , $Z_{ij} \in \{0,1\}$ and it is 1 if the j th customer is assigned to the i th health center.

Liu *et al.* [63] developed a complex cost function to model the location-allocation problem in a urban garbage logistics system, which includes not only the distances between collection centers, transfer stations and landfills but also the cost of locating these infrastructures, the transportation cost for the different waste types, the annual-equivalent cost, etc. The complete formulation can be found in [63]. Their complex formulation is based on (2.20) though they added more terms due to the complexity of the urban garbage system. Neema *et al.* [64] also used a multi-objective location-allocation modeling but they applied it to parks and open spaces location, therefore, their formulation includes terms related with air quality, the land-use and the population coverage instead of the waste transport cost among others.

In location-allocation problem, the demand plays an important role, thus the knowledge about it is crucial. However, usually the demand is not precisely known and the customers are modeled as stochastic functions. This has led several researchers to study the location-allocation problem with fuzzy demand, [65], [66] and [67]. These researchers tried to solve this problem applying different techniques such as (α, β) -cost minimization, [65], which avoids the use of extreme cases of the demand (optimistic or pessimistic) or creating estimators for the solution, [66] and [67], when the demand is a stochastic process.

2.2.3. Summary Location-allocation

In this section we started by presenting the location-allocation problem, when it appeared for first time and what motivated its showing up. Then we exposed the location-allocation problem classification done by Church and we introduced a new class. Finally we reviewed the state of the art of this problem (see Table 2.2)

Table 2.2. Some examples of research about location-allocation.

Year	Researcher	Optimization criteria	Distance	Class
2004	Hsieh and Tien [58]	Minimizing distance	Rectilinear	Median
2004	Li and Yeh [59]	Minimizing distance	Euclidean	Median
2005	Kongsomsaksakul <i>et al.</i> [60]	Minimizing distance	Euclidean	Capacitated
2006	Pang and Feng [61]	Maximizing profit	Euclidean	Competitive
2007	Redondo <i>et al.</i> [68]	Maximizing global profit	Euclidean	Competitive
2007	Wen and Iwamura [65]	Minimizing distance	Euclidean	Capacitated
2007	Yasenovskiy and Hodgson [69]	Minimizing distance	Euclidean	Median
2008	Liu <i>et al.</i> [63]	Minimizing transportation cost	Euclidean	Median
2009	Li <i>et al.</i> [52]	Minimizing travel cost	Euclidean	Capacitated

2010	Li <i>et al.</i> [53]	Minimizing distance	Euclidean	Median
2010	Wen and Kang [67]	Minimizing cost	Euclidean	Capacitated
2010	Sasaki <i>et al.</i> [54]	Minimizing distance	Network-based	
2010	Neema <i>et al.</i> [64]	Minimize several weighted distances	Euclidean	Median
2011	Comber <i>et al.</i> [55]	Minimizing distance	Euclidean	Median

2.3. Optimization

As it has been told in the previous section, LA is an optimization problem as it seeks the best location of N facilities (or services in our case) and the best allocation of the customers to the located facilities. Thus, the search method used is a crucial factor to take into account as it will determine in great measure the goodness of the whole LA method used.

Optimization tries to give the best possible solution for a mathematical problem. Optimization methods differ in how they search the optimum and if they are able to find a global optimum or just a local optimum. Optimization problem was introduced by Fermat, [70], in 17th century. He and Lagrange, [71] and [72], proposed deterministic (calculus-based¹) formulas for finding the optimums. Also, Newton and Gauss, [73] and [74], worked out other optimization methods, but their methods were iterative search² methods that moved towards an optimum. Since then, especially in 20th century, several researchers as R. Bellman [75], R.A. Howard [76], N. Karmarkar [77] or W. Karush [78], have focused their work in the optimization problem.

	Complete	Incomplete
Global search	DFS BFS Backtracking B&B LP	SA GA ACO PSO SO CS FA
Local search		HC GBS TS

Figure 2.9. Optimization methods classification.

¹ Calculus-based methods use the first (and sometimes the second) derivatives in order to find the optimum

² Search methods use values of the target function in order to find the optimum

Optimization methods may be classified in different dimensions such as the type of search and the type of solution they find. The search may be local or global depending on whether the algorithm just explores one path or keeps several paths in memory in order to find a solution [79]. Regarding to the type of solution it may be a sub-optimum or the global optimum. In the literature it is said that the completeness is the algorithm guarantee to find the optimal solution when there is one [79].

Nevertheless, we refer to local search algorithms those that just explore one path or try to improve one initial solution without mechanisms to avoid local optimums and flat regions. Global search algorithms are those that explore several paths to find the optimum, or improve several initial solutions or work with one solutions but they have mechanisms to avoid flat regions and local optimums like simulated annealing. We also refer to complete algorithms those that guarantee the optimal solution and incomplete algorithm those that do not guarantee the optimal solution is found. Figure 2.9 illustrates a 2D classification of the optimization techniques. Note that there are not complete algorithms that perform local search because it is impossible to ensure the optimum is found doing a local search. The methods on Figure 2.9 are explained next.

2.3.1. Complete optimization methods

Complete methods aim to find the optimal solution of a given problem with constraints. The most popular method is *Generate and Test* also known as *Trial and Error* or *Brute Force*. It consists on generating all possible solutions and testing them. Finally, the algorithm chooses the best solution. This method is used when no information is known, as it performs the simplest possible search. So, this method is very cost computing and slow as it has to test all possible solutions.

When the space solution can be represented as a graph, there are several useful search methods to improve the search for the optimal solution. The most popular are:

1. Depth-First Search
2. Breadth-first search
3. Backtracking
4. Branch & Bound
5. A*

There is also another optimization method called linear programming used to solve optimization problems when the search space is a convex polygon delimited by several plains (constraints).

2.3.1.1. Depth-first search

DFS [80] is a method for traveling through or searching a graph. It is a uniformed search method that starts from the root (it first selects one node as the root) and it progresses by expanding the first child node. Then it backtracks and expands the next child node. Therefore it explores the graph by branches, exploring each branch as deeper as possible until the goal node is found or until it hits a node with no children.

DFS can be used for finding bi-connectivity in graphs, finding connected components, topological sorting, etc. It is also used in many fields where people work with hierarchical structures such as power distribution systems [81] or sensor wireless networks [82]. Figure 2.10 illustrates DFS procedure.

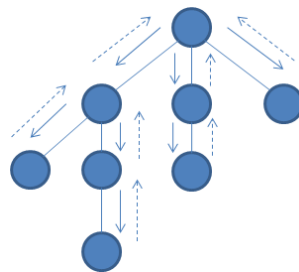


Figure 2.10. Depth-First Search example. Continuous arrows represent the direction of the search and discontinuous arrows represent the backtracks.

2.3.1.2. Breadth-First Search

BFS method, as DFS, is a uniformed search method that instead of exploring the space in levels, it breadth explores the space. Therefore it starts from the root node and then explores all its child nodes (see Figure 2.11). Next, for each of these nodes, it explores their children and so on until all nodes are explored.

BFS can be used for finding all nodes within one connected component, testing bipartiteness¹, finding the shortest path, etc.

BFS has the same time complexity as DFS in the worst case of each one but the space search is much larger than DFS (as DFS only stores a single branch and BFS stores all discovered nodes).

¹ Bipartiteness is a graph property that means the graph can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V . Therefore, U and V are independent sets.

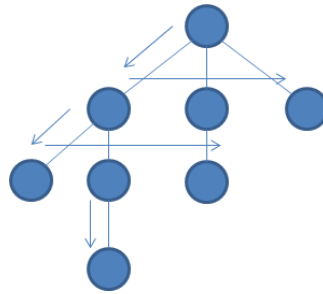


Figure 2.11. Breath-First Search example.

2.3.1.3. Backtracking

Backtracking is a generalization for algorithms that perform a search of the solutions of a computational problem using a DFS approach but in order to reduce the search space, they discard each partial candidate (branch) as soon as it discovers the partial candidate has an inconsistency with the constraints of the formulated problem and it cannot be a valid solution. Then it backtracks and continues with the next candidate.

There are several backtracking algorithms and they introduce some variations to increase the effectiveness of the backtracking algorithm. For example, backjumping allows going up more than one level when the algorithm backtracks; backmarking maintains the information about the last time a variable was instantiated to a value and information about what changed since then and after it uses this information to avoid some consistency checks; constraint learning records new constraints whenever an inconsistency is found in order to reduce, even more, the search space.

Nowadays, several researchers try to work out new backtracking techniques [83] or studying and classifying the vast quantity of backtracking algorithms [84]. Also, some researchers have published important articles about bioinformatics that tell us that cells use backtracking algorithms to establish a relationship between them or for controlling some cellular functions, [85] and [86], proving that backtracking is an intuitive method graph search method also used for several bio-organisms.

2.3.1.4. Branch & Bound

B&B was first proposed by A. H. Land and A. G. Doig in 1960, [87], and it is a search tree pruning technique which procedure is very similar to backtracking. However, B&B is not limited to a unique way of traversing the hierarchy space.

B&B procedure requires two tools:

- Branching: it is a splitting tool used to convert a set of solutions into smaller sets.
- Bounding: it calculates upper and lower bounds to reduce the search space.

The key idea of B&B is that when a tree node is not inside the bounds, so the solutions that come up from this node will be worse than other solutions from other nodes, then it discards this node, thus it prunes the branch that arises from this node.

Some examples of B&B applications are object localization [88], face recognition [89] or scheduling trains in a railway network [90].

2.3.1.5. A*

In 1964 N. Nilsson proposed a heuristic based approach to speed up the Dijkstra's algorithm¹ called A1 [91]. Then, in 1967 B. Raphael made some improvements upon A1 and called this new algorithm A2 [92]. In 1968 P. E. Hart discovered that using a consistent heuristic, A2 was optimal and he called it A* [93].

A* (pronounced A star) is a kind of best-first-search² as it expands the most promising node according to a distance-plus-cost heuristic function $f(x)$. This function is the sum of two terms.

$$f(x) = g(x) + h(x) \quad (2.28)$$

Where $g(x)$ is the path cost function, which is the cost from the starting node to the current node and $h(x)$ is an admissible heuristic estimate³ of the distance to the target node. This estimation can be the Euclidean distance between the node and the goal as it is the minimum distance between them. Thus A* expands the most promising node according to $h(x)$.

Figure 2.12 shows a graph with the distances between nodes and their cost functions and heuristics. In this example, A* would choose node B as the most promising node as $f(b) = 8.4$ and $f(c) = 8.7$. However, $f(d) = 9 > f(c)$, thus A* would expand node C. Then, $f(e) = 9 \leq f(d)$ therefore A* would expand node E until the goal node.

A* method is widely used in robot localization [94], among others. Also all DFS algorithms can be implemented using A* considering a global counter initialized with a very large number. When a node is processed its newly discovered neighbors are assigned to the counter and after each single assignment the counter is decreased by one. Therefore, the earlier a node is discovered the higher its $h(x)$ value.

¹ Graph search algorithm that finds the lowest cost path in a given graph between two nodes expanding the most promising node according to a rule, [79].

² It is a search algorithm which explores a graph by expanding the most promising node. When the first selected path cost is higher than one of the discarded nodes, the algorithm expands the most promising (firstly discarded) node and so on, [79].

³ A heuristic function is said to be admissible if it is not greater than the lowest cost path to the goal, [147].

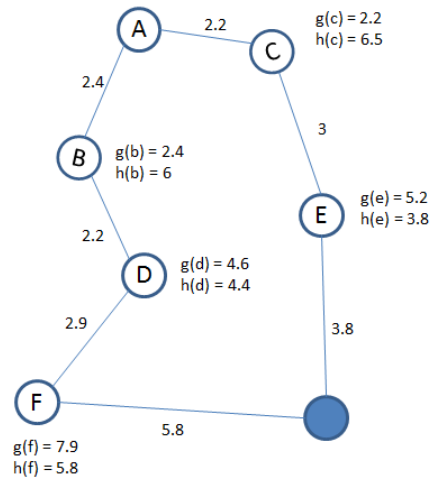


Figure 2.12. Graph example to illustrate A* mechanism

2.3.1.6. Linear programming

The problem of optimizing a linear goal function subjected to linear constraints dates back at least as far as Fourier (1768-1830) when he developed the method called Fourier-Motzkin elimination for solving a system with linear inequalities. The linear constraints are usually posed as linear equalities or linear inequalities and the whole problem can be expressed as

$$\max_x \mathbf{c}^T \mathbf{x} \quad (2.29)$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \quad (2.30)$$

$$\mathbf{x} \geq \mathbf{0} \quad (2.31)$$

Where \mathbf{x} is the vector of variables, \mathbf{c} and \mathbf{b} are vectors with known coefficients and \mathbf{A} is a matrix with known coefficients. The constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ form the feasible region which is a convex polytope¹.

LP is a technique used to solve the optimization problem posed in (2.29), (2.30) and (2.31). The first LP was developed in 1939 by L. Kantorovich [95] though this optimization problem dates back, as said before, some centuries ago. Kantorovich's LP was used in World War II to plan outgoings and returns in order to minimize costs to the army and maximize losses to the enemy. Due to the war situation it was kept in secret. In 1947 George B. Dantzig, [96], published the simplex method which is capable to reduce drastically the number of possible optimal solutions that must be checked which reduces drastically the computational cost. Simplex method first makes a feasible solution at a vertex of the feasible polytope and search the optimal solution going from vertex to vertex of the feasible region with successfully

¹ Geometric object with flat sides, which exists in any general number of dimensions.

higher values of the objective function until it finds the optimal solution or it establishes that no solution exists.

Nowadays linear programming is widely used yet as it has many applications such as decoding and error correcting [97], [98] and [99], compressed sensing [100], minimizing the cost of sensors for complete coverage of a sensor field [101], etc.

2.3.2. Incomplete optimization methods

Sometimes it is not necessary or possible (due to the problem's size) to give the global optimal solution to a given problem, so a local optimum is good enough. Therefore, local search algorithms, which are simpler and less computational costing than the methods presented in the previous section, are better enough for this task. The main local search methods are:

- Hill climbing
- Gradient based search
- Tabu search

Additionally, usually the global optimum is needed but there are some time constraints that discard the use of the complete optimization methods. In answer to this problem, incomplete global search algorithms were developed. These algorithms are able to find an acceptably good solution in a fixed amount of time. The most popular incomplete global search algorithms are:

- Simulated annealing
- Genetic algorithms
- Ant colony optimization
- Particle swarm optimization

Global optimization is an important field of research as researchers are continuously proposing new algorithms. Some examples of the latest optimization algorithms are:

- Spiral optimization
- Cuckoo search
- Firefly algorithm

2.3.2.1. Hill climbing

Hill climbing belongs to the local search algorithms family. It is an iterative algorithm that seeks the optimum by continuously changing an arbitrarily initial solution. It modifies an element of the solution at each iteration and if the change provides a better solution, it performs an incremental change to the new solution. When it finds a local optimum which is not able to avoid by simply changing a single element, the algorithm will consider this optimum as the best possible solution, [79]. Thus it is not able to distinguish between a global optimum and a local optimum.

Despite hill climbing weaknesses, it is a very popular optimizing method due to its simplicity and it may be useful in limited time or real time applications such as online signature verification [102].

Iterated Hill climbing is a hill climbing variant which tries to minimize its dependence on the initial solution (initial conditions) performing a hill climbing on several initial solutions. This reduces the probability the algorithms gets stacked on a plain region or on a local optimum. Nevertheless, it cannot ensure the solution found is the global optimum.

2.3.2.2. Gradient based search

Many methods attempt to use the first (and sometimes the second) order derivatives of the target function. Sometimes a maximum or minimum can be found by solving the equation $\nabla f = 0$, where f is the target function. In many cases, nevertheless, this equation cannot be solved in closed form but the gradient can be computed locally. Therefore, these algorithms seeks the optimum starting from an arbitrary point and going towards the maximum (minimum) through the path with higher (lower) gradient. The most popular deterministic search algorithm is the gradient descend also known as steepest descend method, [103].

When these algorithms find a local optimum, as hill climbing method, they would not be able to avoid it and they would be trapped in it. However, they are widely used, particularly in convex search spaces, due to its fast convergence. Some practical examples of these algorithms are adaptive filtering and signal processing [104].

2.3.2.3. Tabu search

TS was presented in 1989 by Fred W. Glover, [105] and [106]. It is a local search method that iteratively moves from one solution to an improved solution in the neighborhood of the previous one. Thus, it randomly searches the optimum searching the best solution in a neighborhood like hill climbing does. However, Tabu search improves other local search algorithms by introducing a “tabu list” which is a list of the N previously checked solutions and solutions that not satisfy the given constraints. The algorithm uses this list to avoid repeatedly checking these “tabu” solutions.

Tabu search algorithm not only saves a list of the “tabu” solutions, it also can save lists of rules that intend the algorithm to move to promising areas of the search space or intend to unstuck it from some local optimums or *plateau* areas where it has fallen. These rules are introduced in order to avoid the problems of the local search algorithms, although they are not sufficient to guarantee a global search of the optimum whatever the search space is.

2.3.2.4. Simulated annealing

SA is a global search algorithm that was independently described by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi [107] in 1983 and by V. Cerný [108] in 1985. It is based on a metallurgy technique that consists on heating and cooling a material to increase the size of its crystals and reduce its defects. When the material is heated, the atoms are freed from their initial positions (local energy minimums) and wander randomly across the space. Then, the slow cooling gives them more chances of falling in a lower energy state than the initial one. Hence, at each iteration, the SA algorithm considers a neighboring state of the current state, and randomly chooses moving the system to the new state or staying in the current state. This step is repeated until it reaches a good enough state (solution) or it exceeds a given amount of time. The neighboring state generator must be able to avoid local minimums. Thus the performance of the algorithm will depend on how the neighboring state generator is set. If the radius of the neighbor is underestimated, the algorithm may converge to a local optimum. However, if it is overestimated it may slow down the convergence of the algorithm.

Not only the neighbor generator configuration determines the convergence to the global optimum, also the probability of changing the state does. For example, if the energy of state s , $E(s)$, is lower (and so better) than the energy of state s' , $E(s')$, the algorithms will stay in state s and vice versa.

$$P(s | E(s) \leq E(s')) = 1 \quad (2.32)$$

$$P(s' | E(s) > E(s')) = 1 \quad (2.33)$$

This leads the algorithm to the closest sub-optimum. But if we allow the algorithm to do some “bad” moves we spread its mobility and therefore the algorithm performs a “more” global search. For example, if we use the formulation given in [107] (equations (2.34) and (2.35)) we let the algorithm performs some “bad” moves to spread its search.

$$P(s' | E(s) > E(s')) = 1 \quad (2.34)$$

$$P(s' | E(s) \leq E(s')) = e^{\frac{E(s) - E(s')}{T}} \quad (2.35)$$

Simulated annealing is used for 3D face recognition [109], size optimization of energy conversion systems [110] and scheduling [111] among others.

2.3.2.5. Genetic algorithms

The first computer simulations of the evolution started in the 1950s with the work of researchers such as Nils A. Barricelli [112] [113] or Alex Fraser [114] [115], becoming more common in the 1960s. However, it was the work of Ingo Rechenberg [116] and Hans-Paul Schwefel [117] [118] what made artificial evolution a widely recognized optimization method.

Genetic algorithms, as it is known nowadays, became popular through John Holland's book "Adaptation in Natural and Artificial Systems" in 1975, [119]. Although all research done in 1960s and 1970s, genetic algorithms approaches were largely theoretical until 1980s with the increase of desktop computational power available. Thus, in this decade, General Electric sold the world's first commercial GA product for desktop computers, what made possible the implementation of several practical applications of GA algorithms.

GA, as its names suggests, is based on the natural evolution process to perform a global optimum search to a given problem. It belongs to the class of evolutionary techniques, [120], and it is probably its most popular member. GA mimics the natural selection, and the crossover and mutation of the chromosomes that lead natural species evolve.

GA methodology consists on first generating a population of chromosomes, which are candidate solutions to the given problem, and then check the fitness of every chromosome. The fitness quantifies how good is a solution according to a fitness function which usually is the objective function to optimize. After that, it mimics the natural selection, selecting a portion of the initial population that would breed the new generation. The selection process can be random assigning a probability of being chosen to each chromosome according to its fitness or deterministic just choosing the best members.

Once the selection is done, GA mimics the natural reproduction or crossover between chromosomes selecting a pair of parent chromosomes for each child chromosome that is going to be generated. Also the selection of the parents can be done randomly assigning probabilities to every candidate parent according to its fitness and according to the number of children it has generated before in order to keep diversity. The breed of the new chromosome is done choosing genes (solution features) of each parent. There are many types of crossover¹ that differ in the manner they choose the features of each parent. One possibility, called single point crossover, is to enumerate all genes and then generate a random number r . All genes with a lower or equal number than r are from parent one and genes with higher number than r are from parent two. This crossover method can be done using more than one random number generating several intervals. Another popular crossover method is the uniform crossover that consists on assigning a probability to each gene, usually 0.5 to each gene of each parent. And then randomly select them. If the probability is 0.5, the children will have 50% of genes of each parent in average.

After the crossover, a mutation² process is applied in order to keep diversity between generations. Like in crossover, there are several mutation procedures that differ in the probability of mutation p . The most popular are:

- Bit string mutation: it is used when genes are Boolean variables.

¹ Genetic operator used to vary the features of a chromosome(s) from one generation to the next combining the features of the parents.

² Genetic operator used to maintain genetic diversity that consists on altering one or more gene values from its initial state.

Where L is the number of genes of each chromosome.

- Flip bit: it is used when genes are Boolean variables and it consists on applying the NOT Boolean operator to the bit.
- Boundary: it consists on applying lower and upper bounds to all genes. It is used for integer or float genes.
- Uniform: it consists on replacing the value of a chosen gene by a uniform random variable. It is used with integer and float genes.
- Gaussian: it consists on assigning a random Gaussian distributed value to a chosen gene and change its value if the random value exceeds the given lower and upper bounds. This method is used with Boolean genes.

Despite of the previously presented mutation operators there is no rule or hint that tell which is the best operator. It depends on the application and even on the phase of the GA, the diversity of the population. A bad mutation operator can lead to a slow convergence or to stack the algorithm in a local optimum or plateau region.

The selection and reproduction (crossover and mutation) procedures are repeated creating a new population at each iteration. The algorithms ends up when an acceptable good solution (chromosome) is found or it exceeds a fixed amount of time. These constraints make GA being considered an incomplete optimization method as it cannot guarantee the given solution is the optimal. Even more, GA method do not guarantee the optimum is ever found.

GA is a very popular optimization method and it is used in a vast variety of applications among which we highlight clustering [38] [36] and location-allocation problem [59] [55] [54] [63].

2.3.2.6. Ant colony optimization

ACO was first proposed in 1992 by Marco Dorigo in [121] and [122], and it belongs to the swarm intelligence family.

ACO mimics the behavior of ants seeking path between their colony and a source of food. Initially, ants wander randomly until they found food and bring it back to the colony while laying down a pheromone¹ trail. If other ants find such a trail, they will end up wander randomly and they will follow the path to the source food. When they eventually find the food, they will bring it back to the colony while laying down the pheromone, reinforcing the route. The pheromone trail dissipates along the time, thus the shorter path, the shorter time the travelling will be repeated and the higher the pheromone trail. Thus the shorter discovered path will prevail among others. Therefore ACO is a search algorithm that seeks the shorter path between two nodes as the source food is the goal node and the colony the root node.

¹ It is a secreted or excreted chemical factor that triggers a social response in members of the same species

As a short path search algorithm ACO is used, for example, for routing problem in wireless sensor networks [123]. However it is also used for job scheduling [124] or feature selection for classification systems [125].

2.3.2.7. Particle swarm optimization

PSO was first presented by Eberhart and Kennedy [126] in 1995 as an intend to simulate social behavior. Then it was simplified and considered as an optimizer by Shi and Eberhart [127].

PSO algorithm tries to achieve the optimal solution to a given problem, iteratively improving a set of candidate solutions. It starts with a population of “particles”, called swarm, that wander randomly around the search space. At each iteration, the algorithm checks the fitness of the positions (solutions) found by each particle (particle’s positions). Let us assume \mathbf{p}_i the best position found by the i th particle and \mathbf{g} the best position found by the whole swarm. If the new position occupied by the i th particle is the best position found by the particle, \mathbf{p}_i is updated to this new position. Additionally, if one of the new positions is better than \mathbf{g} , \mathbf{g} is updated. Then the algorithm guides the particle’s move assigning to each particle a velocity according to the following expression:

$$\mathbf{v}_i = \lambda \mathbf{v}_i^r + \varphi_p r_p (\mathbf{p}_i - \mathbf{x}_i) + \varphi_g r_g (\mathbf{g} - \mathbf{x}_i) \quad (2.36)$$

$$r_p, r_g \sim U(0,1) \quad (2.37)$$

$$\mathbf{v}_i^r \sim U(\mathbf{v}_{low}, \mathbf{v}_{up}) \quad (2.38)$$

Where \mathbf{v}_i^r is a uniform random variable bounded by \mathbf{v}_{low} and \mathbf{v}_{up} , \mathbf{x}_i is the current position of the particle and λ , φ_p and φ_g are parameters selected by the user to control the behavior and efficacy of the algorithm as they control how it guides the particles to the local optimums found. There has been a lot of research about the parameters of the PSO as they may have a great impact on the optimization performance [128] [129] [130] [131].

Since PSO presentation in 1995, it has experienced many changes as researchers have derived new versions for new research fields such as power systems [132] or new applications such as work scheduling [133].

2.3.2.8. Latest optimization approaches

As stated previously, there is a great interest on finding new optimization algorithms that improve the performance of the existing methods. Among the latest optimization we want to highlight SO, CS and FA.

One of the most recent approaches is SO [134] [135] developed by Tamura and Yasuda in 2011. It is a metaheuristic¹ algorithm inspired by the spiral phenomena in nature. The algorithm sets some initial points (solutions) in the search space and checks them in order to find the best solution found. Then it sets the best solution as the center of the spiral and moves the solutions towards this center with a spiral movement. At each iteration the center is re-estimated between all new solutions found.

CS, [136], is based on some cuckoo² species that lay their eggs in other host birds nest in order that the owner of the nest take care of their children with the risk that if the host discover the eggs are not its own, it will either throw these alien eggs away or simply abandon its nest. Thus in Cuckoo search each cuckoo egg represents a possible that is laid in a nest (the number of nests is fixed previously). The nest with better eggs will carry over the next generation. There is also a discovering operator that mimics the behavior of the host bird when it discovers the eggs are not their own. This operator usually works on the worsts nests and it is a probability of dumping the eggs in the nest.

FA, [137], is inspired by the flashing behavior of fireflies. Fireflies use their flash ability in order to attract other fireflies. Thus, firefly algorithm sets a number of fireflies (candidate solutions) with an assigned brightness according to their fitness with the goal function. Fireflies will move towards the brightest firefly. However, the perceived brightness decreases as the distance between fireflies increases which makes that not all fireflies will feel attract by the same firefly. If one firefly does not find any other brighter firefly it will wander randomly.

2.3.3. Summary optimization

In this section we reviewed how optimization algorithms can be classified. Then we introduced a new definition for completeness and global search and we classified the most relevant algorithms according their completeness, pointing out the type of search they perform and their main features. Finally, Table 2.3 summarizes main features of each algorithm presented in this section.

¹A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure the information in order to find efficiently near-optimal solutions, [148].

² Medium sized slender birds of the family of Cuculidae.

Table 2.3. Optimization methods.

Algorithm	Completeness	Search	Other features
Brute Force	Complete	Global	Computational cost proportional to the number of candidate solutions.
Depth-First search	Complete	Global	Worst case computational cost $O(E + V)$. Worst case space complexity $O(V)$.
Breadth-First search	Complete	Global	Worst case computational cost $O(E + V)$. Worst case space complexity $O(V)$. Larger search space than DFS.
Backtracking	Complete	Global	Increases the search speed allowing backtracks. Performance depends on the algorithm and the application.
Branch & Bound	Complete	Global	Reduces the computational cost pruning the infeasible solutions Performance depends on the estimations.
A*	Complete	Global	Reduces the computational cost expanding the most promising node. Performance depends on the heuristic.
Linear programming	Complete	Global	Faster than the previous algorithms. Only useful with problems modeled by linear equations.
Hill climbing	Incomplete	Local	Low computational cost. Good performance in convex search spaces. Target function does not have to be differentiable.
Iterated Hill climbing	Incomplete	Local	Performs several local searches. Low computational cost. Better solutions than Hill climbing. Good performance in convex search spaces. Target function does not have to be differentiable.
Gradient based search	Incomplete	Local	Low computational cost. Very fast convergence. Good performance in convex search spaces. The target function has to be differentiable.
Tabu search	Incomplete	Local	Low computational cost. Target function does not have to be differentiable. Increase the search space making a tabu list with infeasible solutions and previously found bad solutions.
Simulated annealing	Incomplete	Global	Usually finds good solutions with a few iterations. Usually it is not capable to improve the solutions found in the early iterations. Target function does not have to be differentiable.
Genetic algorithm	Incomplete	Global	Target function does not have to be differentiable. Usually gives better solutions when more time is given. Slower than simulated annealing and other optimization algorithms. The fitness functions may generate bad chromosomes.
Ant colony optimization	Incomplete	Global	Efficient for traveling salesman problem. Positive feedback accounts for rapid discovery of good solutions. Theoretical analysis is difficult. Sequences of random decisions. Time to convergence uncertain though it is guaranteed.
Particle swarm optimization	Incomplete	Global	Target function does not have to be differentiable. Easy calculations. Cannot work out the problems of non-coordinate systems.
Spiral optimization	Incomplete	Global	Target function does not have to be differentiable. Cannot work out the problems of non-coordinate systems.
Cuckoo search	Incomplete	Global	Target function does not have to be differentiable. More robust results than basic PSO. Slow convergence.
Firefly algorithm	Incomplete	Global	Target function does not have to be differentiable. Cannot work out the problems of non-coordinate systems.

2.4. Optimization methods applied to location-allocation problem

LA problem is a combinatorial problem that can be solved using complete or incomplete optimization methods. However, the method must have some acceleration or space bounding techniques in order to find a solution feasibly because the location problem of deciding the best match over k possible matches for n facilities, becomes k^n possible combinations. Just as example, solving this problem for 3 possible matches and 20 bars becomes 3486784401 possible combinations, and when there are 5 possible matches and 40 bars becomes $9.09 \cdot 10^{27}$ possible combinations.

Table 2.4. Summary of Location-Allocation latest research.

Year	Researcher	Optimization criteria	Location model	Optimization method
2004	Li and Yeh [59]	Minimizing distance	Median	GA
2005	Kongsomsaksakul <i>et al.</i> [60]	Minimizing distance	Capacitated	GA
2006	Pang and Feng [61]	Maximizing profit	Competitive	CSA ¹
2007	Redondo <i>et al.</i> [68]	Maximizing global profit	Competitive	GA
2007	Wen and Iwamura [65]	Minimizing distance	Capacitated	Linear programming & GA
2007	Yasenovskiy and Hodgson [69]	Minimizing distance	Median	CPLEX
2008	Liu <i>et al.</i> [63]	Minimizing transportation cost	Median	GA
2009	Li <i>et al.</i> [52]	Minimizing travel cost	Capacitated	GA
2010	Li <i>et al.</i> [53]	Minimizing distance	Median	PSO
2010	Sasaki <i>et al.</i> [54]	Minimizing distance		GA
2010	Neema <i>et al.</i> [64]	Minimize several weighted distances	Median	GA
2011	Comber <i>et al.</i> [55]	Minimizing distance	Median	GA

¹ Clonal Selection Algorithm is an optimization method that mimics the acquired immunity of biological systems. It consists on a combination of GA without recombination operator and hill climbing.

Additionally, the LA search space is not a convex space what discards the use of methods like linear programming. Also due to the several local optimums that may exist in the search space local optimization methods such as hill climbing or gradient based methods are not expected to provide good solutions. Therefore metaheuristic methods such as genetic algorithms or particle swarm optimization as they perform a global search may provide good solutions though they do not guarantee the given solution is the optimum. Table 2.4 shows some researches about LA and the optimization algorithms the authors have chosen to solve the problem.

2.5. Summary

In this chapter we have reviewed the state of the art on the techniques related to this master's thesis. First, we provided a rough history of clustering and then we introduced a brief description of clustering and its traditional main techniques such as k-means, hierarchical clustering, fuzzy clustering, etc. We also presented some techniques that use tools such as neuronal networks or genetic algorithms to perform clustering. Additionally, a description of some of the latest clustering techniques is given. All of them have been classified according to hard or soft clustering and we have also explained their main strengths and weaknesses.

Next we described the location-allocation including four LA problems. We also provide a state of the art of the mathematical modeling of this problem and some fields where it is common.

Finally, we introduced the need of optimization algorithms to solve problems like LA. Following we roughly classified optimization methods as complete optimization methods and incomplete optimization methods and we described the main techniques in each class. Finally we provide some information about which optimization methods are more used to solve LA problems.

CHAPTER 3. Clustering

The immobile LA problem is a k^n combinatorial problem where k is the number of possible services (matches) and n the number of facilities. The applicability of optimization methods is tied up to the dimensionality of the problem. Therefore, one way to deal with the dimensionality is to partition the space and finding the solution for each subspace. The global solution can be achieved then combining the partial solutions. Obviously, the partition of the space will strongly determine the result of the optimization process.

In this Master's Thesis the LA problem refers to the problem of deciding the sport events a group of bars should broadcast in order to maximize the amount of people that will go to such bars to watch the matches. In this case, therefore, the facilities are the bars that broadcast a certain sport event, and the possible locations are all bars. The group of bars can be all bars in a region, or a country or in the world (we used a large list of Spanish bars). Thus the problem of dimensionality has to be dealt with.

The most important feature that determines if a customer prefers to watch a match in a bar instead of another one is the distance, especially when it is large. It is obvious that nobody is going to travel 100 km just to see a match in a common bar. Thus it seems that clustering the data space according to the geographical positions of the bars could be a good way to convert a huge problem in several smaller problems.

Among the clustering techniques presented in Chapter 2 we selected the following: k-means, Lloyd's algorithm, region growing, hierarchical clustering, genetic algorithm based clustering and affinity propagation. We discarded fuzzy clustering techniques because we seek a hard clustering as we do not want bars in different clusters. Competitive learning has been discarded due to its need to be trained. Density clustering also does not fit to our problem because it deals with isolated elements as outliers, obviating them, what is not desired. Finally multi-way clustering is useful when there are two or more features to take into account to perform clustering; nevertheless, we want to make a clustering using only the coordinates of each element what makes multi-way clustering useless.

All clustering results have been obtained using the Euclidean distance among the bars. Clustering results have been analyzed using two clustering quality indices (Calinski index and Davies-Bouldin index), the size and number of the clusters and the elapsed time.

3.1. K-means

As it has been told in subsection 2.1.1, k-means is the oldest and one of the most popular clustering methods due to its simplicity. One of its most important weaknesses is it needs to know the number of clusters a priori. This is an important problem as the number of clusters is not known a priori in our case. One way to minimize this drawback is to run several times the algorithm for different number of clusters and finally choose the best clustering according to the quality indices. To determine the clustering quality we use the Calinski index. Algorithm 3.1 shows the resulting algorithm. Steps 3 to 10 of the algorithm consists of the basic k-means algorithm as explained in subsection 2.1.1, but with some special treatment to deal with empty clusters as explained below. The remaining steps, 1-2 and 11-19, evaluate the algorithm according to the results obtained with the Calinski index. The main drawback of this algorithm is in dense regions more centroids would be set, since centroids are randomly settled and it will divide dense homogeneous regions into several different clusters (see Figure 3.1).

Algorithm 3.1. K-means algorithm (with empty clusters removed)

Require: $N_{executions}, N_{iterations}$

1. **for** $execution = 1$ **to** $N_{executions}$
 2. Choose the number of clusters K randomly
 3. Select randomly K objects as centroids
 4. **for** $iteration = 1$ **to** $N_{iterations}$
 5. Assign every bar to the closest centroid
 6. **if** $cluster_k$ is empty
 7. Remove $cluster_k$
 8. **else**
 9.
$$c_k = \frac{\sum_{j=1}^{N_k} x_j}{N_k}$$
 10. **end if**
 11. Calculate Calinski index CI'
 12. **if** $CI' > CI$
 13. $CI = CI'$
 14. Save clustering and remove the previous one.
 15. **else**
 16. Exit the loop
 17. **end if**
 18. **end for**
 19. **end for**
-

On the other hand, the use of the native k-means, so the centroids are initially placed randomly across the data space, solves this problem while causing a second one: the presence of empty clusters. To avoid this problem there are two alternatives:

- Remove empty clusters as in Algorithm 3.1. This reduces drastically the number of clusters due to the large amount of empty clusters.
- Re-assign centroids (Algorithm 3.2), so that empty clusters are moved randomly to other place of the space (see Figure 3.2). In that case the number of partitions remains the same.

K-means tends to provide circular clusters (see Figure 3.1 and Figure 3.2) as it uses the distance between objects and cluster centroids to decide where to put each object. Quantitative results are provided in Table 3.1.

Algorithm 3.2. K-means algorithm (with cluster reassignment)

Require: $N_{executions}, N_{iterations}$

1. **for** $execution = 1$ **to** $N_{executions}$
 2. Choose the number of clusters K randomly
 3. Select K geographical coordinates randomly (cluster centroids)
 4. **for** $iteration = 1$ **to** $N_{iterations}$
 5. Assign every bar to the closest centroid
 6. **if** $cluster_k$ is empty
 7. Assign it a new random centroid c_k
 8. **else**
 9. $c_k = \frac{\sum_{j=1}^{N_k} x_j}{N_k}$
 10. **end if**
 11. Calculate Calinski index $newCI$
 12. **if** $newCI > oldCI$
 13. $oldCI = newCI$
 14. Save clustering and remove the previous one.
 15. **else**
 16. Exit the loop
 17. **end if**
 18. **end for**
 19. **end for**
-

Table 3.1. K-means results

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
k-means (Algorithm 3.1)	578	28955.66	0.717	896	27	1	59
k-means (Algorithm 3.2)	1170	50166.93	0.499	444	74	1	1001

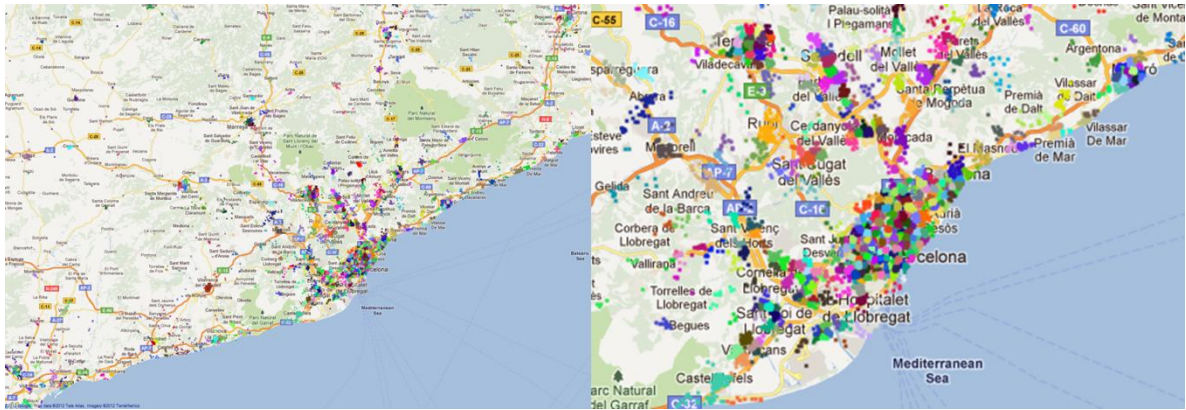


Figure 3.1. Clustering result using Algorithm 3.1. The number of clusters in the dense regions is higher.

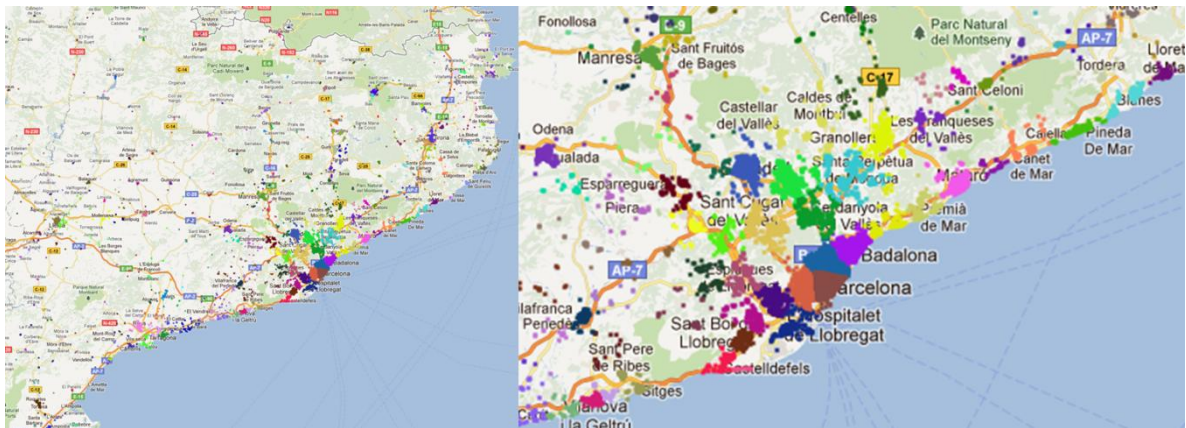


Figure 3.2. Clustering result using Algorithm 3.2.

Both algorithms use the same number of iterations (30) to converge a single initialization, i.e. both algorithms use the same number of iterations for the part of Algorithm 2.1. Additionally both are executed the same number of times (30) using different initializations (number of clusters and seeds positions) Despite of that, Algorithm 3.1 is two times faster than Algorithm 3.2. However Algorithm 3.2 provides a much better result. This comes from the initializations where Algorithm 3.1 plants the seeds in the same position of some elements of the dataset randomly selected. This makes that dense regions will have more seeds than less dense regions. This provokes that dense regions are separated with several clusters. Instead, Algorithm 3.2 plants the seeds randomly over the whole data space avoiding this problem. In conclusion, Algorithm 3.2 provides good clustering results as Davies-Bouldin and Calinski indices say (see Table 3.1) and a smaller number of clusters than Algorithm 3.1.

Figure 3.3 shows how the k-means clustering quality evolves depending on the number clusters. There we can see that the optimum is reached between 400 and 500 clusters what matches with the results presented in Table 3.1. Thus, we can conclude that the presented Algorithm 3.2 is able to find a nearly optimal k-means clustering.

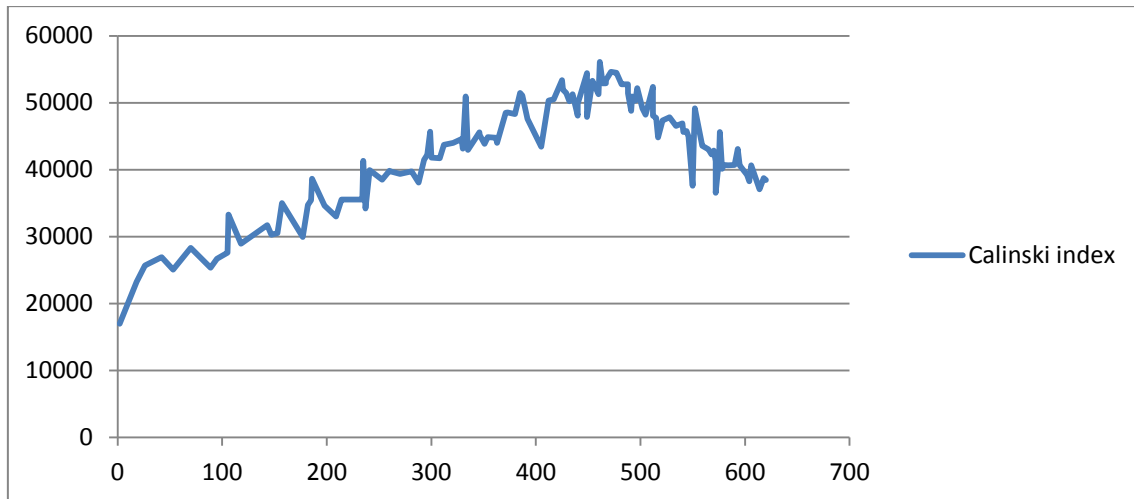


Figure 3.3. Calinski index vs. number of clusters using Algorithm 3.2

3.2. Lloyd's algorithm

Lloyd's algorithm is a k-means algorithm approximation. It also needs the number of clusters k but it does not randomly decide the initial position of the K centroids. Instead of it, it groups the data in k random groups and computes the centroid of each group. Then it reassigns bars to the closest cluster and recalculates cluster centroid as k-means does. This modification avoids the problem of empty clusters provided by isolated centroids.

Table 3.2. Lloyd's algorithm results

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
Lloyd's algorithm (Algorithm 3.3)	395	21958.88	0.698	17	1	137	3423

Results on the Lloyd's algorithm are provided in Table 3.2. As it is showed, Lloyd's Algorithm is faster than the two k-means algorithms. The convergence threshold is 30 iterations and the number of executions done is 30 as Algorithm 3.1 and Algorithm 3.2. It tends to find much less clusters than k-means algorithms, between 10 and 20 times less. And, obviously, the clusters found are larger. Calinski index indicates the quality of the clustering is lower than both k-means algorithms. However, Lloyd's algorithm clustering seems visually better than Algorithm 3.1 result (see Figure 3.4) as it does not divide dense and homogeneous regions in several different clusters.

This is corroborated by Davies-Bouldin index which is better for Lloyd's algorithm clustering. Its tendency to join isolated and distant elements in the same cluster penalize the Calinski index result.

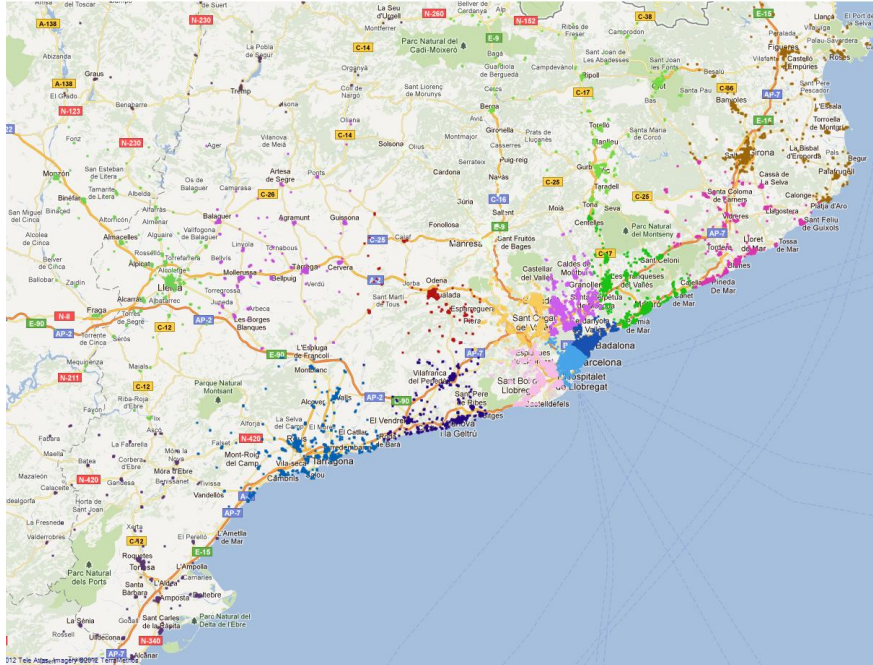


Figure 3.4. Clustering result using Algorithm 3.3.

Algorithm 3.3. Lloyd's algorithm

Require: $N_{executions} = 5$, $N_{iterations} = 30$

1. **for** *execution* = 1 to $N_{executions}$
 2. Choose the number of clusters K randomly
 3. Divide randomly the dataset into K groups
 4. Compute the centroid c_k of every group
 5. **for** *iteration* = 1 to $N_{iterations}$
 6. Assign every bar to the closest centroid
 7.
$$c_k = \frac{\sum_{j=1}^{N_k} x_j}{N_k}$$
 8. Calculate Calinski index *newCI*
 9. **if** *newCI* > *oldCI*
 10. *oldCI* = *newCI*
 11. Save clustering and remove the previous one.
 12. **else**
 13. Exit the loop
 14. **end if**
 15. **end for**
 16. **end for**
-

Another Lloyd's algorithm result is that it tends to equilibrate the population in each cluster, as it initially divides the dataset in several parts, what makes that clusters in dense regions occupy smaller areas and clusters in less dense regions occupy larger areas.

Figure 3.5 shows the Calinski index of different clustering using different number of clusters. There we can see that the clustering quality decreases (with the used dataset) as the number of clusters increases. This matches with the optimum Lloyd's clustering presented in Table 3.2.

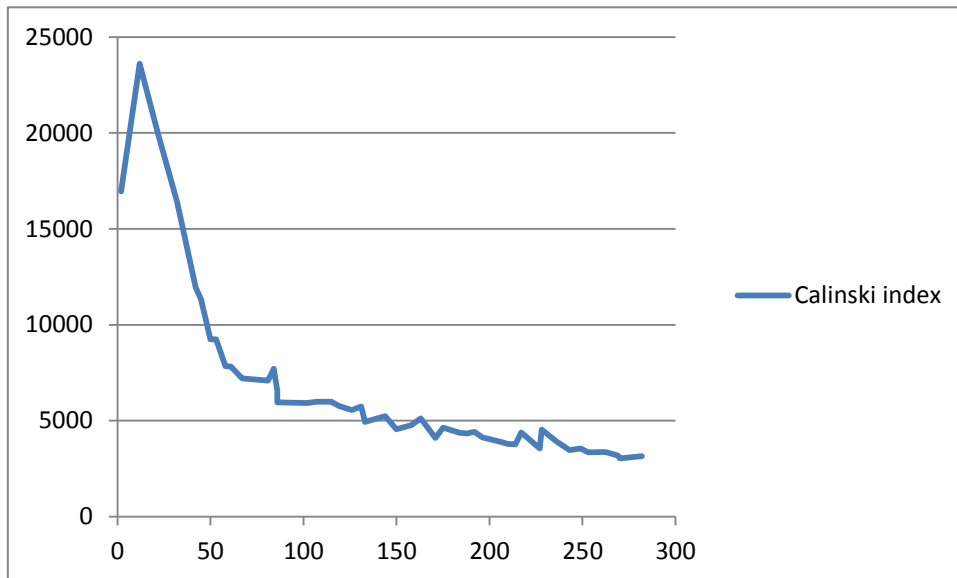


Figure 3.5. Calinski index vs. number of clusters using Lloyd's algorithm

3.3. Region Growing

Region growing is does not tend to find circular shaped clusters as it expands clusters without taking into account the centroids of the clusters. Region growing needs a threshold that tells it when to stop expanding a region. This parameter can be used as a parameter of control that allows the user to decide the minimum distance between two bars of the same cluster.

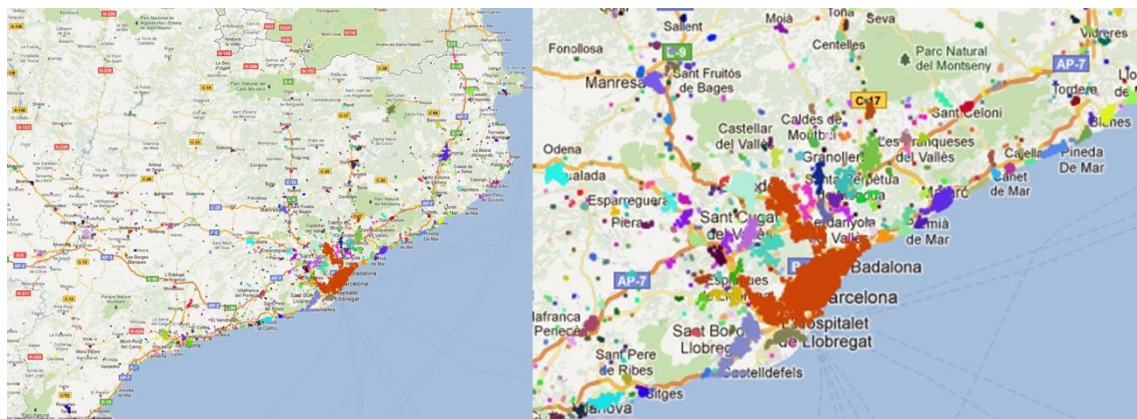
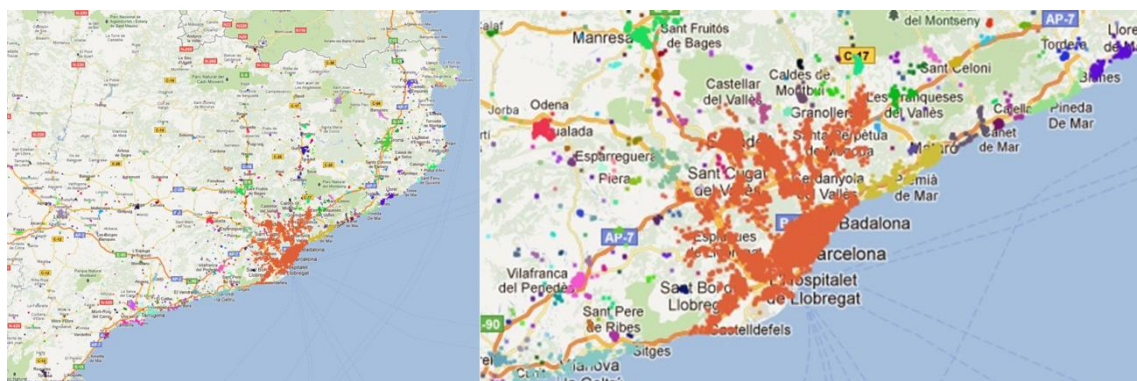
Algorithm 3.4. Region Growing

Require: D_{max}

1. Assign to each object all objects closer than D_{max}
 2. Create a list of non-assigned objects $List_{ObjLeft}$
 3. **while** $List_{ObjLeft}.Count > 0$ **do**
 4. $object = List_{ObjLeft}[0]$
 5. Create a new cluster and add $object$, its neighbors, the neighbors of the neighbors and so on
 6. Remove assigned objects from $List_{ObjLeft}$
 7. **end while**
-

Table 3.3. Region growing results

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
Region growing (Algorithm 3.4) $D_{max} = 0.5\text{km}$	3	5800.08	0.265	1935	1114	1	4467
Region growing (Algorithm 3.4) $D_{max} = 1\text{km}$	6	2614.59	0.228	1095	521	1	5885
Region growing (Algorithm 3.4) $D_{max} = 2\text{km}$	12	1182.52	0.224	707	288	1	8202
Region growing (Algorithm 3.4) $D_{max} = 5\text{km}$	37	430.88	0.383	280	93	1	10733

Figure 3.6. Clustering result using Region Growing and $D_{max} = 1\text{km}$ Figure 3.7. Clustering result using Region Growing and $D_{max} = 2\text{km}$

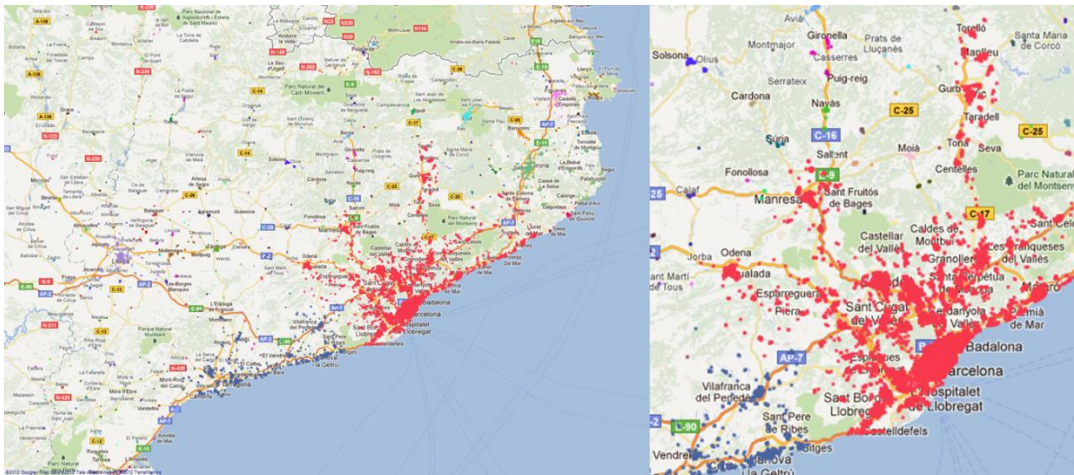


Figure 3.8. Clustering result using Region Growing and $D_{max} = 5\text{km}$

Region growing is the fastest algorithm of those described in this chapter (see Table 3.3 and Table 3.7), especially when the distance threshold is lower than 15 or 20 km as it is speeded up when it uses small clusters.

As said before, Region growing does not measure the distance between cluster centroids and the elements, even more it does not care where the cluster centroids are. This makes the cluster to not have circular forms what reduces Calinski index. Nevertheless, Region growing results have the lowest Davies-Bouldin index what indicates that it performs a good clustering though different than the other techniques. For this reason, we cannot use Calinski Index to compare region growing results. For example, results on Table 3.3 show that region growing provides a better Calinski index for lower D_{max} , however, the results are not better according Davies-Bouldin index. This fact means that for lower D_{max} the clusters found by region growing are more circular shaped and, therefore, the Calinski index is greater.

Another feature of region growing is the number of elements of the largest cluster. This largest cluster has between 30% and 60% of the elements in the dataset (see Figure 3.6, Figure 3.7 and Figure 3.8). This result tells us that there is a region with a high density of elements. This feature may put out some problems because our initial aim is to divide the initial dataset in several smaller dataset to perform separate LA in each one and if one sub-dataset has about half the elements, this partition will not help us.

Also, region growing puts every isolated element in a different cluster, what makes that it provides several clusters with only one element. That is not a problem but at the same time, as said before, it puts together in the same cluster a vast amount of elements.

3.4. Agglomerative hierarchical clustering

Hierarchical clustering is a clustering method completely different from k-means and its variants. This technique does not require any parameter as it starts assigning one object per cluster and then joins the closer clusters and so on. However it needs some parameters to decide when to stop joining clusters. An important measure to decide when to stop the algorithm is comparing the jump between two consecutive joins, i.e. when the algorithm decides to join two clusters because they are very close to each other, it is said the algorithm makes a jump as there is an increase of the minimum diameter of the clusters.

We define a jump as the 5% of difference between hierarchy levels ($Th_{jump} = 5\%$) as it is the value for which we achieved the best results regarding results on Figure 3.9. Moreover, we set an extra threshold called d_{min} to avoid that the algorithm stops in the earlier iterations when the size of the clusters is very small. Algorithm 3.5 provides implementation.

Hierarchical clustering's main weakness is its speed. As it can be seen in Table 3.4, the amount of time it needs to finish is huge and unfeasible. In terms of clustering quality, it achieves good results as Calinski and Davies-Bouldin indices show though it does not provide the best clustering. Beyond indices, the clusters it finds are well defined as the algorithm detects well the main centroids (centroids of the main regions). The number of clusters it finds is similar to region growing, and lower than k-means algorithms. But as said before, the unfeasible computational time it needs avoids its use for our purposes.

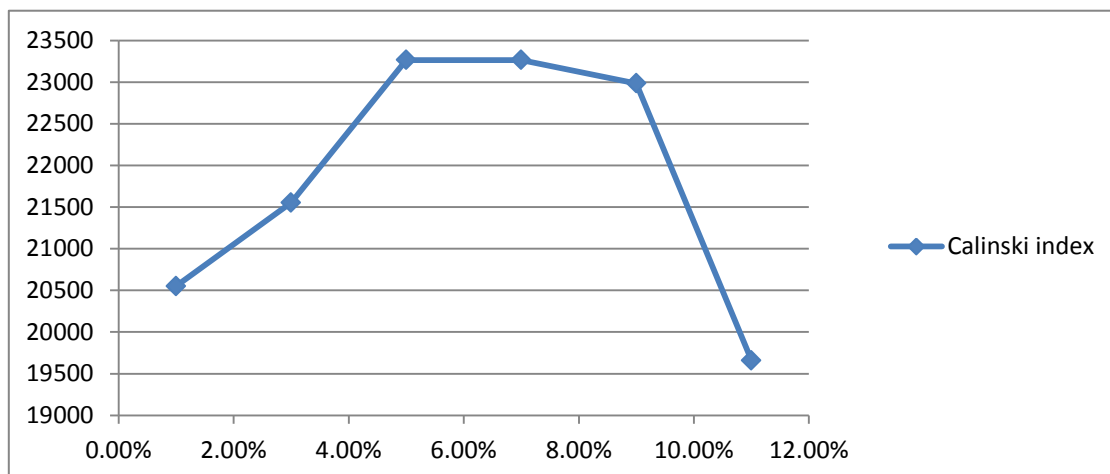
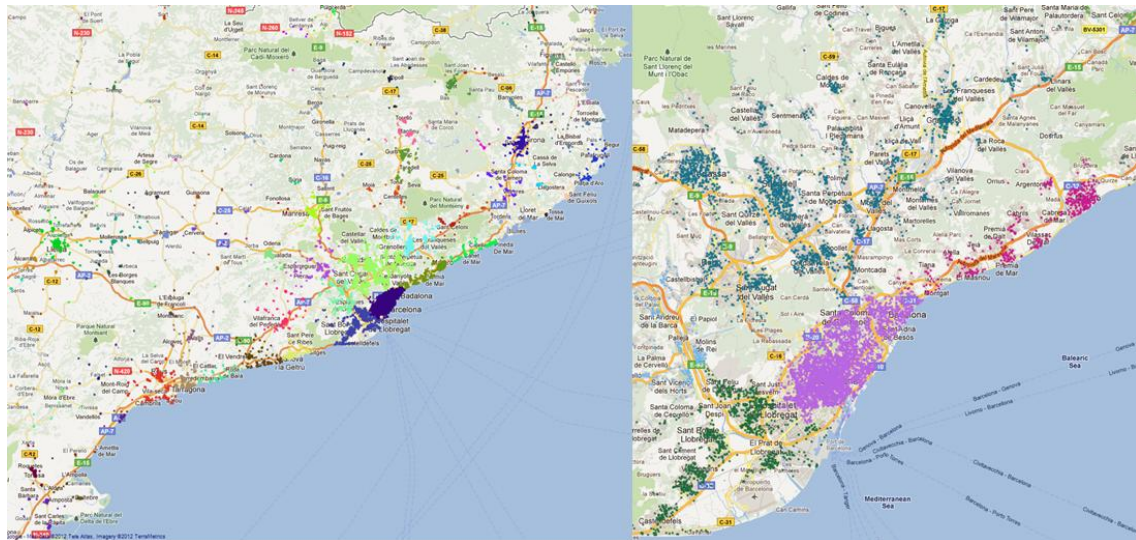


Figure 3.9. Hierarchical clustering results using different Th_{jump}

Algorithm 3.5. Agglomerative hierarchical clustering**Require:** $Th_{jump} = 0.05$ and $Th_{dmin} = 10km$

1. Create one cluster per object
2. **while** $N_{clusters} > 1$ **do**
3. Calculate distances between clusters
4. Find the minimum distance d_{min}
5. Join the pair of closer clusters
6. $jump = \frac{d_{min} - d_{old}}{d_{min}}$
7. **if** $jump > Th_{jump}$ && $d_{min} > Th_{dmin}$ **do**
8. Save the new clustering
9. Exit while loop
10. **end if**
11. **end while**

**Figure 3.10.** Clustering result using hierarchical clustering.**Table 3.4.** Hierarchical clustering results

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
Hierarchical clustering (Algorithm 3.5)	36636	16592.55	0.472	139	10	1	4487

3.5. Genetic algorithm based clustering

Genetic clustering exploits the searching capability of genetic algorithms for automatically evolving the number of clusters as well as proper clustering of any data set. We followed the approach [138] resulting in Algorithm 3.6.

This algorithm starts creating a population of chromosomes (initialization, steps 1-6). Each chromosome has a random number of clusters centroids K_i . However the chromosome lengths are equal because it sets a constant length $L \geq K_i, \forall i$ and then it randomly distributes the centroids across this string of L slots. It fills the empty slots with a constant symbol that indicates the slot is empty (see Figure 3.11).

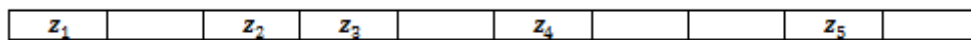


Figure 3.11. Chromosome example. The $K_i = 5$ centroids are distributed along the $L = 10$ slots

To select the parents of one population to generate offspring to the next generation roulette selection is applied. It consists in generating a number of copies of every individual proportional to their fitness. All these copies are put in a pool and then some of them are selected randomly to breed a new generation (steps 13-15).

Fitness (step 11) of a chromosome is related to the clustering quality resulting from the centroids it represents. Thus, the fitness is calculated using the Davies-Bouldin Index (DBI) so that the lower DBI the better the clustering (see subsection 2.1.11). So, the fitness is defined as $1/DBI$. We discarded the use of Calinski index to define the fitness because it provides results with high differences, not proportional to the differences between clusterings. This fact provokes that the diversity of the population decreases a lot in few iterations.

Regarding crossover (step 16), a single point crossover is used. It consists of randomly selecting a number p between 1 and L , where L is the length of the chromosome. Then two children are created assigning to the first child, the first p genes of the first parent and the last $L - p$ genes from the second parent. The second child is created assigning the first p genes from the second parent and the last $L - p$ genes from the first (see Figure 3.12).

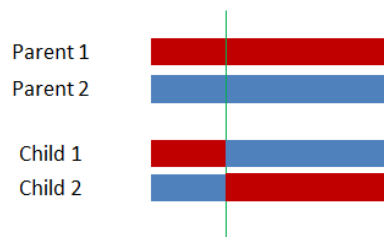


Figure 3.12. Single point crossover

Regarding mutation (step 17), each centroid $\mathbf{z}_{k,i}$ of each chromosome, with probability $\mu_{mutation}$, is changed according the following rule:

$$\mathbf{z}_{i,k} = \begin{cases} \mathbf{z}_{i,k} \cdot (1 \pm 2\delta) & \mathbf{z}_{i,k} \neq 0 \\ \pm 2\delta & \mathbf{z}_{i,k} = 0 \end{cases} \quad (3.1)$$

Where + or – signs occurs with equal probability and δ is a uniform random number between 0 and 1.

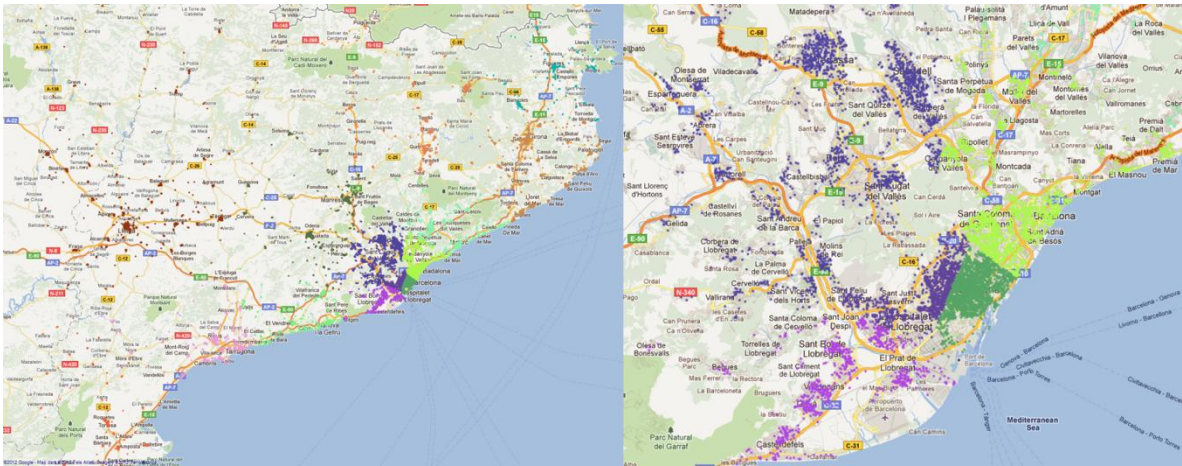


Figure 3.13. Genetic clustering

Genetic clustering is a slow algorithm as the results show on see Table 3.5. It also tends to find a low number of clusters like Lloyd's algorithm. In fact, it provides similar results as Lloyd's algorithm, so it is able to find good results in terms of Calinski and Davies-Bouldin indices, though not the best. As Lloyd's algorithm it tends to equilibrate the amount of elements in each clustering varying the cluster's areas (see Figure 3.13). This property is useful as we want to partition a huge problem into several smaller problems. Nevertheless, genetic clustering tends to divide a dense and homogeneous region in a few clusters what is not desired.

The number of generations analyzed is 100; the population size is [25,150] (other parameters are expound in Algorithm 3.6). Table 3.5 shows the results obtained using different population sizes. There we can see that the larger population the algorithm has more chances to find a better solution, though it does not ensure it finds the best solution. Moreover, the amount of memory needed increases with the population, thus we finally limited the population size to 75 individuals. Figure 3.14 shows the best individual fitness in each generation. When the algorithm is close to 100 generations it starts to stabilize it.

Table 3.5. Genetic clustering results

Population size	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
25	2187	13008.09	0.776	28	1	134	1201
50	2910	9854.67	0.845	27	1	157	1256
75	1188	16889.09	0.536	5	1	350	11181
100	4575	15911.56	0.757	14	1	366	2305
125	2010	16142.77	0.618	3	1	1849	11746
150	5122	18620.14	0.591	5	1	366	10594

Algorithm 3.6. Genetic clustering

Require: $Max_{generation} = 100$, $\mu_{crossover} = 0.8$, $\mu_{mutation} = 0.01$,
 $Size_{population} = 75$

1. **for each** chromosome i **in** the population
2. Generate a number K_i
3. Choose K_i points randomly from the data (objects positions)
4. Distribute these points randomly in the chromosome
5. Set unfilled positions to *null*
6. **end for**
7. **while** $generation < Max_{generation}$
8. **for each** chromosome i **in** the population
9. Extract the K_i centers stored in it
10. Perform clustering by assigning each object to the closest cluster (centroid)
11. Compute fitness using Davies-Bouldin index
12. **end for**
13. Create a pool of chromosomes making N_i copies of each chromosome and N_i proportional to the chromosome fitness
14. **for** $j = 1$ **to** $N_{children} / 2$
15. Select randomly two parent chromosomes from the pool
16. Perform crossover to create two new chromosomes
17. Perform mutation over the new chromosomes
18. **end for**
19. Select the $Size_{population}$ best members (between parents and children) for the next generation
20. $generation = generation + 1$
21. **end while**
22. Select the best member in the last generation

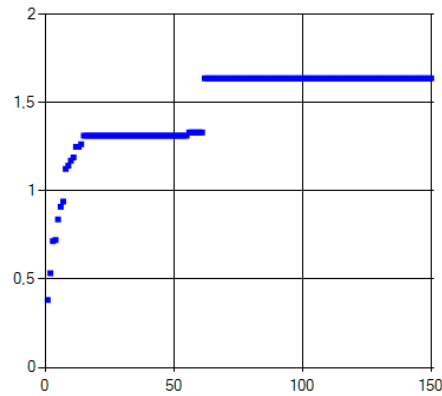


Figure 3.14. Fitness evolution (vertical axis) of the best individual vs. generation (horizontal axis)

3.6. Affinity propagation

Affinity propagation is a clustering technique developed by Frey and Duek, members of the University of Toronto. They have published some implementation of their algorithm in C and Matlab. The implementation, we had access to, is done in Matlab.

Affinity propagation tends to discover the more representative objects in a dataset, so it can perform clustering without additional parameters. However, it uses $N \times N$ matrices, where N is the number of objects in the dataset. This implies the algorithm needs a huge amount of memory resources when it works on large datasets. The Matlab implementation provided by the University of Toronto can run over datasets smaller than 16000 objects. For larger datasets a sparse version of affinity propagation should be used.

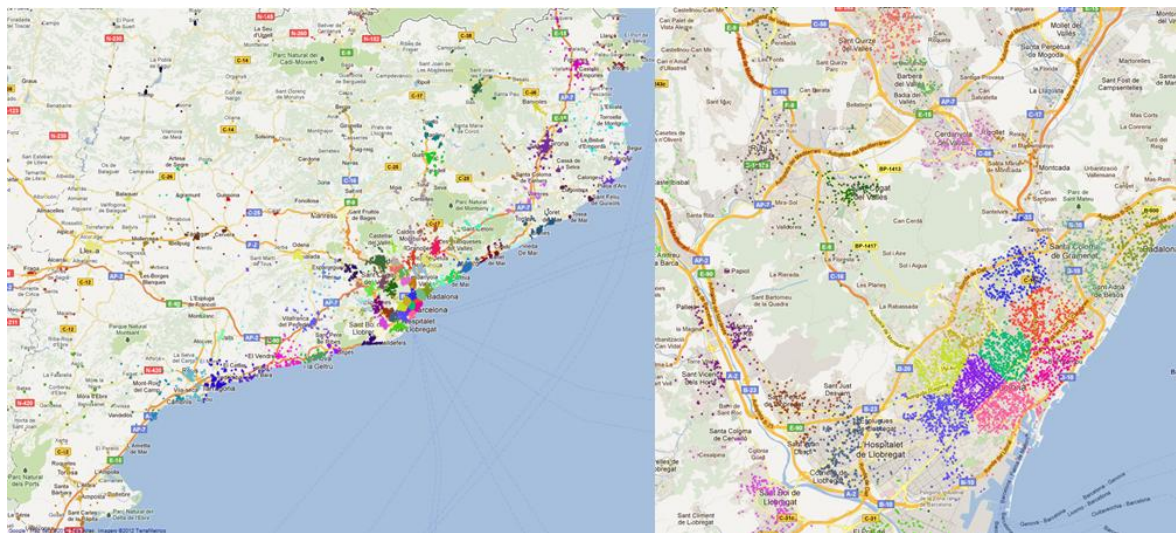


Figure 3.15. Clustering result using Affinity propagation

As shown in Table 3.6, the time expended by affinity propagation's Matlab implementation is 3892 seconds, so it is a slow algorithm when it works over large datasets. However the clustering quality in terms of Calinski and Davies-Bouldin indices is good, though k-means (Algorithm 3.2) provides better results and in a lower amount of time.

As Lloyd's algorithm or genetic clustering, affinity propagation tends to equilibrate the elements of each cluster varying the area each cluster occupies. However, it finds smaller clusters than Lloyd's algorithm and genetic clustering, what induces it to find more clusters than these two algorithms. It also divides a dense homogeneous region into a few different clusters.

Table 3.6. Affinity propagation results

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of min. clusters	Smallest cluster	Largest cluster
Affinity propagation	3892	27037.92	0.665	92	1	18	690

3.7. Discussion

In this chapter we analyze the behavior of some clustering techniques in our LA problem, obtaining several clusters of bars placed in Catalunya. Following a comparison of these techniques is presented (Table 3.7).

All techniques presented before, except region growing, tend to create circular shaped clusters as they use the distance between objects and cluster centroids. However, region growing uses the distance between objects, and it does not matter where the cluster centroids are. Therefore, the use of indices such as Calinski or Davies-Bouldin may bias the comparison between the beforehand mentioned techniques because these indices use cluster centroids to evaluate the within (clusters) homogeneity and the between (clusters) heterogeneity. Despite of that, they are a useful tool that we have used.

If we focus in the quality indices to evaluate the different clustering techniques, it can be said that k-means technique provides the best performance followed by genetic clustering and Lloyd's Algorithm. The differences between the results achieved using Algorithm 3.1 and Algorithm 3.2 come from the way the initial seeds are planted. The first one plants a random number of seeds selecting a random amount of objects. This induces to put more seeds around the densest regions what implies that dense regions will be divided in several clusters. This may reduce the quality of the clustering as it tends to divide homogeneous regions. Instead, Algorithm 3.2 plants seeds uniformly across the data space avoiding the previously exposed problem. Also, reassigning a new centroid to the empty clusters is like the algorithm is

executed more times what tends to improve the final solution but also slow down the algorithm.

Lloyd's algorithm gives similar results as k-means does, however the quality is lower. This is caused, mainly, for the algorithm tendency to join isolated objects because it starts doing random groups of objects, and although it recalculates the centroids and reassign objects to the closest centroid, some centroids are set between isolated objects. A proof of that is that the less dense areas the clusters are bigger, they occupy wider areas, and the densest regions are divided in several clusters. This feature tends to equilibrate the amount of elements in each cluster. Genetic clustering and affinity propagation also have this feature.

Genetic clustering provides good results, though it does not improve the results given by Lloyd's algorithm, either k-means algorithms or affinity propagation. It is also slower than these algorithms. This fact means that the genetic operations work worse than simply rerun the algorithm with different initializations.

Hierarchical clustering does not require any parameters to perform clustering. However, it needs some thresholds to decide when to stop joining clusters. Depending on these thresholds different results will be obtained. Generally, hierarchical clustering provides good results, but the spent time is really huge and the quality of the result does not compensate it, what makes hierarchical clustering useless in front of other algorithms.

Affinity propagation provides very good results in terms of Calinski and Davies-Bouldin indices. As Lloyd's algorithm or genetic algorithm, it tends to group the less dense regions in a few wide clusters and divides the more populated areas in some small but dense clusters. This reduces the differences between clusters in terms of population. Despite the similarities with Lloyd's algorithm results, affinity propagation clustering is better than Lloyd's, but also spends much more time.

Region growing does not provide clustering results with good Calinski indices, although it achieves good Davies-Bouldin indices. The reason may be, Calinski indices evaluate de global heterogeneity between clusters and the homogeneity into the clusters and Davies-Bouldin index evaluates the quality using equation (3.2) (see page 30).

$$\frac{1}{K} \sum_{i=1}^K \max_{j, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{d_{ij,t}} \right\} \quad (3.2)$$

So, it only cares the pairs of cluster that maximize the within differences divided by the distance between the cluster centroids. Note that the within differences are computed using the cluster centroids, thus the more circular the cluster is the better. This makes Davies-Bouldin index introduce a bias in the quality evaluation of non-circular clusters. But in our case, it tells us region growing results are not bad. To evaluate the quality of the region growing results we can focus in how the clusters are and how many there are. Region growing gives a similar amount of clusters (depending on the distance threshold) as k-means algorithms, although there is a big difference between the sizes of the clusters (see the largest cluster given by $D_{max} = 1\text{km}$ has 5885 objects and the smaller 1). This is caused for the nature of the

data (there are a very dense large region and a lot of isolated objects) but also for how region growing works. It guarantees that the frontiers between clusters will be wide empty regions and the wideness is determined by the distance threshold. This is not guaranteed for the other presented algorithms. Therefore region growing provides a completely different clustering than other algorithms and the fitness of it depends on the desired clustering features. Another point for region growing is its speed. It is the fastest algorithm what makes it the only feasible algorithm with larger datasets (the dataset used has 15578 objects). However it has a weakness as it tends to join different dense regions due to the outliers between them. In the dataset used there some dense regions very close to each other and most of the elements belong to them. This causes that region growing finds a large and dense cluster, which contains most of the elements in the datasets.

Table 3.7. Clustering algorithms results achieved from a dataset of 15578 Catalan bars.

Algorithm	Expended time (s)	Calinski Index	DB Index	Number of clusters	Number of minimal clusters	Smallest cluster	Largest cluster
k-means (Algorithm 3.1)	578	28955.66	0.717	896	27	1	59
k-means (Algorithm 3.2)	1170	50166.93	0.499	444	74	1	1001
Lloyd's algorithm (Algorithm 3.3)	395	21958.88	0.698	17	1	137	3423
Region growing (Algorithm 3.4) $D_{max} = 1\text{km}$	6	2614.59	0.228	1095	521	1	5885
Region growing (Algorithm 3.4) $D_{max} = 2\text{km}$	12	1182.52	0.224	707	288	1	8202
Region growing (Algorithm 3.4) $D_{max} = 5\text{km}$	37	430.88	0.383	280	93	1	10733
Hierarchical clustering (Algorithm 3.5)	36636	16592.55	0.472	139	10	1	4487
Genetic clustering (Algorithm 3.6)	4575	15911.56	0.757	14	1	366	2305
Affinity propagation	3892	27037.92	0.665	92	1	18	690

Summarizing, region growing is the fastest algorithm and it is the one that fits best when there is a large amount of objects. However, in our case, we wish the clusters be circular and to minimize the differences between clusters in terms of population. Thus the best algorithm is k-means (Algorithm 3.2). If the dataset is very large, a k-means approximation like Lloyd's algorithm should be used due to time issues.

CHAPTER 4. Location-allocation

This Master's Thesis studies the problem of global optimization applied to location-allocation. Particularly, it tries to find the best solution to the problem of where to broadcast sport events (location) and how to allocate the audience (allocation). Figure 4.1 shows a LA example; on the left it shows the position of the customers (small colored squares) and their desired match (each match is represented with a color) and the positions of the bars (big black squares); on the right it shows the same customers and bars but after LA, so it shows the match assigned to each bar that maximizes the global weighted demand.

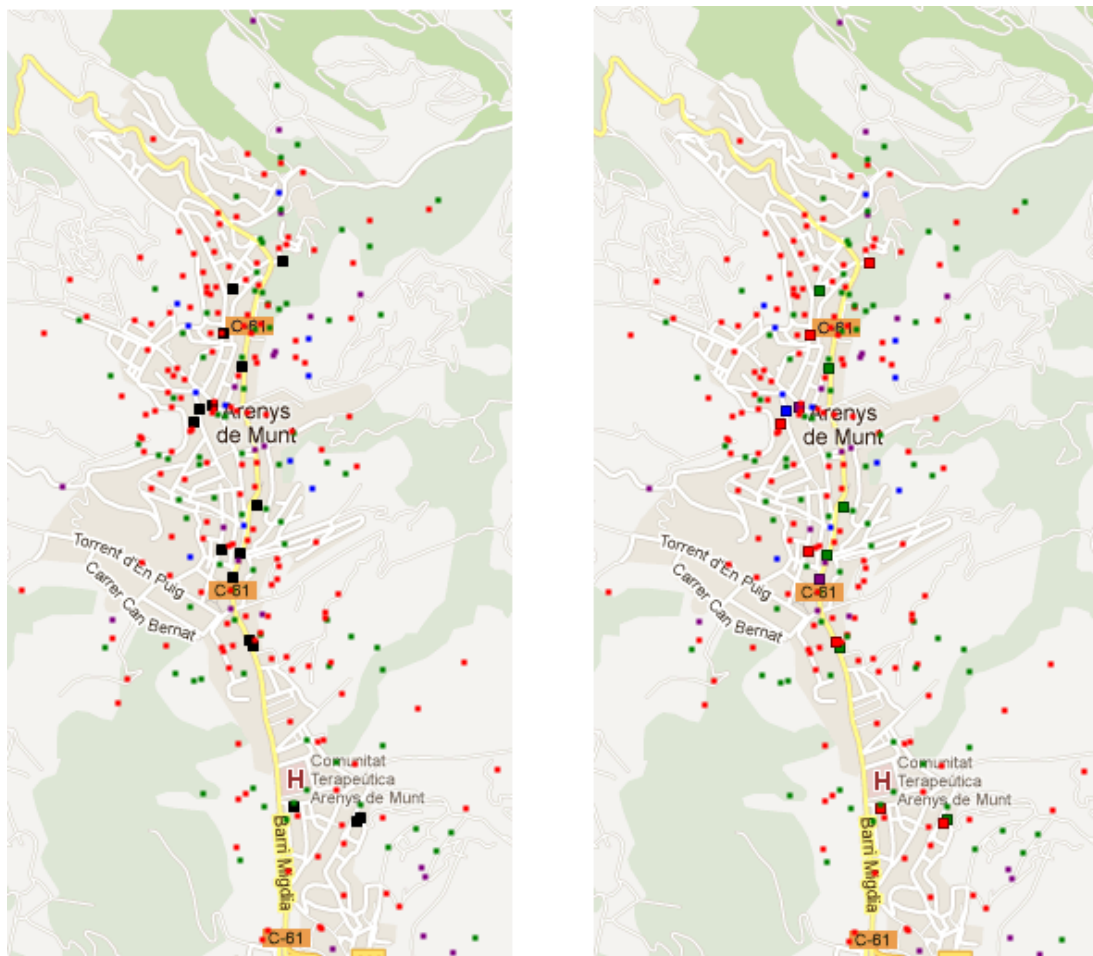


Figure 4.1. Example of LA.

In this chapter we firstly present the mathematical model we used to solve this problem; then we provide our implementation of GA, SA and CS to solve the LA problem; finally we present the results obtained with the three algorithms in order to analyze their performance on solving the stated LA problem. The three optimization algorithms are applied to 10 different clusters of different shapes and sizes that are showed in Appendix B. We compare the solutions of the three algorithms in terms of fitness, equation (4.6), in terms of time needed to find it and in terms of percent of satisfied customers and percent of empty bars.

In section 2.3 we presented several optimization methods, nevertheless, we discarded most of them as they are useless for the features of our problem. For example, we discarded brute force, DFS, BFS and backtracking for the size of the graph. If there are only 40 bars and 4 possible matches the number of nodes is 2560000. We also discarded other path search algorithms (A*, B&B and ACO) as the space search they need is very huge.

Moreover, due to the solution space is a non-convex space with several local optimums, algorithms such as linear programming, gradient based search, tabu search or hill climbing are not expected to obtain good solutions.

Additionally, the solution space belongs to a non-coordinate system what presents the possibility of using algorithms that search the optimum solution using a coordinate system such as PSO, FA or SO.

On the other hand, GA, SA and CS are heuristic methods that perform a global search and do not need a coordinate system. However, they are not complete algorithms but the more time they have, the better the solution they find. Thus, these optimization methods are suitable to find, at least, a good enough solution to our LA problem.

4.1. Mathematical model

Notations

$\mathbf{x}^q = \langle x_1^q, x_2^q, \dots, x_{N_{bars}}^q \rangle$ the q th solution vector. It contains the match each bar broadcasts.

$x_i^q \in [0, \dots, N_{matches}]$ the match assigned to the i th bar in the q th solution.

$M_j \in [0, \dots, N_{matches}]$ the desired match of the j th customer.

$z_{ij}^q \in \{0, 1\}$ a variable that indicates if the j th customer is assigned to the i th bar in the q th solution.

d_{ij}^2 the square Euclidean distance between the j th customer and the i th bar.

There are several factors that take part when a person decides to watch a sport event and where he or she goes to watch it. We focus on the people that go to a bar to watch a certain sport event. These audience or customers, decide to go to a certain bar depending on if it broadcasts the match, depending on the distance to the bar, and depending on some preferences like tradition, food, etc. Nevertheless, we assume that the main factor that takes part in the bar choice by a customer, between several bars that broadcast the same match, is the physical distance. We approximate the distances between bars and customers by the Euclidean distance because it can be easily calculated since the positions of customers and bars are known.

Then, the problem of deciding the match each bar has to broadcast in order to maximize the global demand can be formulated as

$$\max_{\mathbf{x}^q} \left\{ \sum_{i=1}^{N_{bars}} s(\mathbf{x}_i^q) \right\} \quad (4.1)$$

Where

$$s(\mathbf{x}_i^q) = \sum_{j=1}^{N_{customers}} \frac{z_{ij}^q}{1 + d_{ij}^2} \quad (4.2)$$

Subject to

$$\forall i \quad \sum_{j=1}^{N_{customers}} z_{ij}^q \leq C_i \quad (4.3)$$

$$\forall j \quad \sum_{i=1}^{N_{bars}} z_{ij}^q \leq 1 \quad (4.4)$$

$$\mathbf{x}_i^q \neq M_j \rightarrow z_{ij}^q = 0 \quad (4.5)$$

The target function considers the distance between the bars and the customers as it is the main factor that influences in the customers bar choice. Thus, we defined $s(\mathbf{x}_k(i))$ as equation (4.2), where $N_{customers}$ is the number of customers and d_{ij}^2 is the square distance between the i th bar and the j th customer. Equation (4.3) is used to avoid overflowing the capacity of each bar.

Note that this new definition of the problem does not consider as mandatory to satisfy all customers, it just maximizes the total number of customers weighting them with the distance to the assigned bar.

4.2. Genetic algorithm

GA is an optimization technique that uses the ability of genetic operators to iteratively improve a population of solutions in order to achieve the best possible solution (see subsection 2.3.2.5).

We solve the proposed LA problem using the GA, Algorithm 4.1. We defined the chromosomes as strings of length L , where L is the number of bars of the cluster under study. These strings contain, in each slot, the match each bar has to broadcast. Thus each chromosome only has the information of which match is assigned to each facility.

Regarding the reproduction process (step 6), the chromosomes are combined using a single point crossover (see Figure 3.12) and a mutation function where each slot has a probability $\mu_{mut} = 0.01$ of being mutated, thus of changing the match for another arbitrary one. When a slot is changed, all matches have the same probability of being selected. Regarding the crossover, the pairs of parents are selected according the roulette selection (see subsection 2.3.2.5).

To maintain the population size, a reinsertion operator (step 9) has been used after each new generation breeding. This operator selects the best members between the parents and the children according to their fitness and then it inserts them to the next generation. The fitness of each chromosome is the result obtained using equation (4.6) after allocating customers to their closest bar that broadcasts their desired match. Thus, the fitness value consists of the global weighted demand.

$$Fitness(q) = \sum_{i=1}^{N_{bars}} \sum_{j=1}^{N_{customers}} \frac{z_{ij}^q}{1 + d_{ij}^2} \quad (4.6)$$

In order to work out the most suitable population size for GA we have run it with different population sizes (see Table 4.1 and Table 4.2). Given the results, we can say that for larger populations the achieved solution is usually better (higher fitness), although the time needed (as well as the memory resources) to find it is also greater. So there is a compromise between the quality of the solution and the cost to find it. We have considered that a population size of 50 individuals is an appropriate value.

Algorithm 4.1. Genetic algorithm based location-allocation**Require:** $N_{chromosomes} = 50$, $MaxGenerations = 100$, $MaxUselessGenerat = 30$

1. $G = 0$; $UG = 0$; $F_{old} = -\infty$
2. Initialize: create $N_{chromosomes}$ new chromosomes
3. Perform allocation of customers for each chromosome
4. Compute the fitness of each chromosome
5. **while** ($G < MaxGenerations$) & ($UG < MaxUselessGenerat$)
6. Create $N_{chromosomes}$ new chromosomes using crossover and mutation operators
7. Perform allocation of each chromosome
8. Compute the fitness of each chromosome
9. Select the best $N_{chromosomes}$ for the next generation
10. Find the best chromosome and its fitness F_{max}
11. **if** $F_{max} < F_{old}$
12. $UG = UG + 1$
13. **else**
14. $UG = 0$
15. $F_{old} = F_{max}$
16. **end if**
17. $G = G + 1$
18. **end while**

Table 4.1. Fitness of the final solution found by GA using different population sizes

Population size	Number of facilities					
	8	18	48	50	72	127
5	81.82	199.53	737.95	788.81	1128.17	1975.65
10	82.66	197.77	750.44	810.24	1136.55	1977.84
25	83.65	199.98	760.38	822.99	1139.24	1986.52
50	83.61	200.50	757.23	818.76	1144.07	1985.82
100	83.65	201.92	759.06	826.63	1142.85	1984.38
150	83.65	201.76	761.23	821.06	1145.83	1991.16

Table 4.2. Elapsed time (s) by GA with different population sizes

Population size	Number of facilities					
	8	18	48	50	72	127
5	0.038	0.213	1.381	2.165	3.022	7.454
10	0.056	0.304	3.524	3.439	6.625	21.242
25	0.169	0.730	9.165	9.632	18.948	52.218
50	0.394	1.822	21.424	23.238	40.701	101.783
100	0.696	4.163	42.833	46.181	85.762	223.688
150	1.008	6.631	64.651	66.839	122.257	289.406

4.3. Simulated annealing

SA, as said in subsection 2.3.2.4, is a global search technique based on a metallurgy technique that heats and cools the material to move their atoms in order they reach lower energy states. SA tries to iteratively improve an initial solution with this heating-and-cooling process. The algorithm has three main steps: neighbor selection, energy calculation and the state choice.

Regarding the neighbor selection (step 13 in Algorithm 4.2), SA usually selects a neighbor solution of the current solution in order to compare both solutions and then move the algorithm to the new solution if this is better. This neighbor selection needs to define a coordinate system or some kind of neighborhood function. As in our case we do not have any coordinate system, we implemented a neighborhood function that builds a new solution (neighbor) from another one by assigning to each bar a certain probability of changing its match. The probability is determined according an exponential function that depends on the occupation of the match (see equation (4.7)), where the occupation o_i of each bar is the number of customers of this bar divided by its capacity. Thus, if the bar i has low occupation when it broadcasts a certain match, it will have a greater probability that in the new solution (neighbor) it has to broadcast a different match. So, in this new neighbor function we implicitly defined a proximity criterion that depends on the occupation of the bars, hence when a bar is full it is very difficult that the neighbor solution selected by the function changes the match of these bar because a solution with a different match for this bar is a very “far” solution.

$$P(\text{change the match}) = e^{-\frac{o_i}{\tau}} \quad (4.7)$$

Additionally, we defined the parameter τ to scale the probability depending on the ratio $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i}$ (steps 1-5) and depending on the phase of the algorithm (steps 10-12) to increase the convergence in the final last iterations. Figure 4.2 shows the probability functions given different τ .

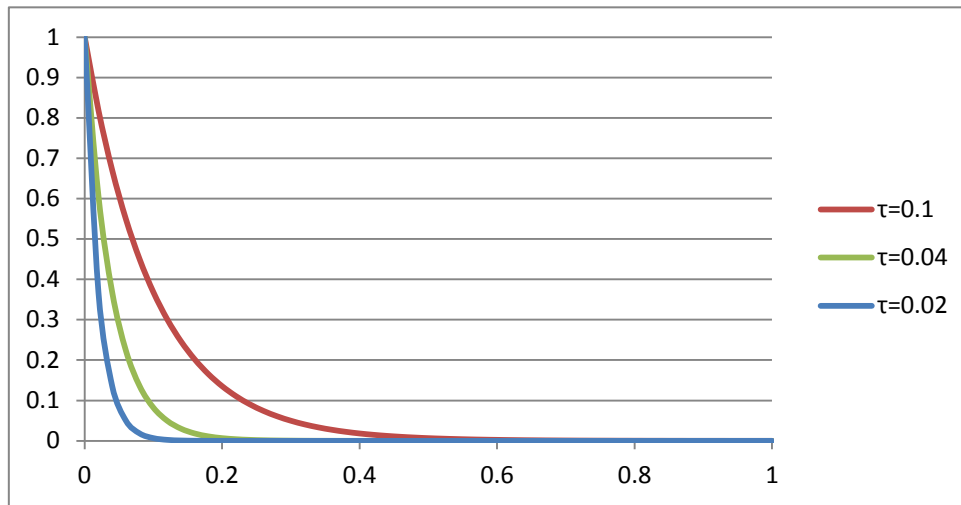


Figure 4.2. Probability function of changing a match given the occupation of the bar

Despite we have finally chosen an exponential probability function with a variable τ , we tested different probability functions with different τ to make a final choice. Precisely we compared four cases:

- Exponential probability function with a variable τ and τ takes the values showed in Algorithm 4.2.
- Exponential probability function with a constant τ and $\tau = 0.05$. In this case the probability of changing a match also depends on the occupation of the bar in question but τ has always the same value.
- Variable uniform probability function. The probability of changing a match does not depend on the occupation of the bar in question, however, this probability is variable depending on the ratio $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i}$ and depending on the phase of the algorithm. Thus, if $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i} > 0.3$ then the probability of change the match is 0.1, if $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i} < 0.3$, then it is 0.04 and if the algorithm is in its last 25% iterations, then it is 0.02.
- Constant uniform probability function. In this case the probability of changing a match is always the same, 0.05.

These four cases have been considered in order to work out the factors that should (and should not) take part in the neighbor selection.

Regarding the energy calculation (steps 7 and 14) we compute the energy as

$$E(q) = \sum_{i=1}^{N_{bars}} \sum_{j=1}^{N_i} \frac{z_{ij}^q}{1 + d_{ij}^2} \quad (4.8)$$

Where N_{bars} is the number of bars, N_i is the number of customers of the i th bar and d_{ij}^2 is the square distance between the i th bar and its j th customer. As said in equation (4.1), the algorithm has to maximize the energy, so in our case it seeks the highest energy state or the solution with higher energy. Note that the target function is the same as genetic algorithm uses.

After selecting a neighbor solution and computing its energy, SA has to decide to move to the new state (solution) or to stay in the current state (steps 16-27). This decision is made according equations (4.9) and (4.10), where s and s' are the current and the new state respectively, $E(s)$ and $E(s')$ their respective energies and T the temperature. So if the new state's energy is higher, the algorithm moves to the new state (steps 16-18) and if the new state's energy is lower, there is a small probability to move to the new state (steps 23-26); otherwise it stays in the current state.

$$P(s' | E(s) < E(s')) = 1 \quad (4.9)$$

$$P(s' | E(s) \geq E(s')) = e^{-\frac{E(s') - E(s)}{T}} \quad (4.10)$$

To compare the performance using the different neighborhood functions we used the energy (global weighted demand) E , the number of allocated customers and the number of bars with a low occupation (we considered low occupation those lower than 4% of the capacity). Table 4.3 shows the results obtained after performing SA in ten different cases (ten different clusters) using the four types of neighborhood functions. We can see that the neighborhood functions that use an exponential probability depending on the occupation of the bar obtain better results than the functions that use a probability that does not depend on the occupation of the bars. Moreover, the functions that change the probability depending on the ratio $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i}$ and depending on if the algorithm is in its last iterations, also obtain better results than those that do not. Thus, we can say that if the occupation of the bars, the ratio $\frac{N_{customers}}{\sum_{i=1}^{N_{bars}} C_i}$ and the iteration number of the algorithm take part in the selection of a neighbor solution outperform the algorithm. In fact, the neighborhood function that combines all these factors obtains the best results (see column 1 Table 4.3).

Algorithm 4.2. Simulated annealing based location-allocation**Require:** $T_a = 0.0001, T = 1, \delta T = 0.99$

1. **if** $N_{customers} / \sum_{i=1}^{N_{bars}} C_i < 0.3$
2. $\tau = 0.04$
3. **else**
4. $\tau = 0.1$
5. **end if**
6. Select an initial random solution s
7. Compute the energy E
8. $E_{best} = E$
9. **while** $T_a < T$
10. **if** $T < 10T_a$
11. $\tau = 0.02$
12. **end if**
13. Select a neighbor solution s'
14. Compute the energy of the new solution E'
15. $L = E - E'$
16. **if** $L < 0$
17. $E = E'$
18. $s = s'$
19. **if** $E_{best} < E$
20. $E_{best} = E; s_{best} = s$
21. **end if**
22. **else**
23. Select a uniform random number x between 0 and 1
24. **if** $x < e^{-\frac{L}{T}}$
25. $E = E'; s = s'$
26. **end if**
27. **end if**
28. $T = \delta T \cdot T$
29. **end while**

Table 4.3. LA results using SA with different neighborhood functions. Best results are in bold face

Exponential probability with variable τ			Exponential probability with $\tau = 0.05$			Variable uniform probability			Constant uniform probability		
E	% of allocated customers	% of bars with occupation < 4%	E	% of allocated customers	% of bars with occupation < 4%	E	% of allocated customers	% of bars with occupation < 4%	E	% of allocated customers	% of bars with occupation < 4%
217.04	95.33	0	211.34	94.00	0	214.45	95.00	0	216.15	93.00	0
104.43	97.82	1	103.85	98.55	3	103.04	98.55	2	104.01	96.38	3
1223.49	99.43	0	1218.94	98.93	0	1221.93	98.93	0	1218.18	98.93	2
616.49	99.86	3	616.55	100	3	614.95	99.86	5	613.67	99.86	6
2010.62	100	0	2013.74	100	1	2005.71	100	8	2007.23	100	13
996.03	100	12	994.11	100	11	993.98	100	19	991.81	100	23
5579.03	99.83	1	5571.28	99.71	3	5535.93	99.73	48	5531.09	99.68	41
2622.78	99.86	20	2622.36	99.89	28	2612.07	99.96	89	2606.94	99.75	91

4.4. Cuckoo search

Cuckoo search is an optimization technique that emulates the behavior of some cuckoo species when they lay their eggs in foreign nests in order that other birds take care of the cuckoo eggs (see subsection 2.3.2.8). Thus, the cuckoo search algorithm generates a certain number of initial solutions and drops them in the *nests*, and then it continuously generates new solutions and drops them in the *nests*, though only one solution per nest is allowed, so it automatically removes the worst solution. Algorithm 4.3 shows our cuckoo search implementation for solving LA problem.

Cuckoo search, as simulated annealing, needs a neighborhood function in order to select new neighbor solutions in order to find a better solution. So, due to the good results the neighborhood function achieved using the simulated annealing algorithm (see section 4.3), we decided to use this function in the cuckoo search as well. Thus, to generate a neighbor solution (step 13), the algorithm uses an exponential probability function that depends on the occupation of the bars. Our algorithm also use a parameter that is modified according to the ratio $N_{customers} / \sum_{i=1}^{N_{bars}} C_i$ (steps 1-5) and the iteration number (steps 9-11).

The fitness function (steps 7 and 14) used in this algorithm is the same used in genetic algorithm and simulated annealing.

In order to find the appropriate number of nests for Algorithm 4.3, we have run it with different number of nests. The obtained results are presented in Table 4.4. There we can see that the presence of more nests does not improve the quality of the solution found. Moreover, in contradistinction to GA, the use of more nests do not increase the elapsed time, since in each iteration just one solution is generated, although it increases the memory resources as it has to keep more solutions. Thus, we could choose any value (though 2 is the worst) but we have finally chosen 10 nests.

Algorithm 4.3. Cuckoo search based location-allocation**Require:** $N_{nests} = 10$, $Iterations = 1000$

1. **if** $N_{customers} / \sum_{i=1}^{N_{bars}} C_i < 0.3$
2. $\tau = 0.04$
3. **else**
4. $\tau = 0.1$
5. **end if**
6. Generate N_{nests} new random solutions and drop each solution into a nest
7. Compute the fitness of each solution
8. **for** $i = 0$ **to** $Iterations$
9. **if** $i > 0.75Iterations$
10. $\tau = 0.02$
11. **end if**
12. Select a random nest j and its solution s_j
13. Generate a neighbor solution s' from solution j
14. Compute the fitness of s'
15. Select a random nest k and its solution s_k
16. **if** $fitness(s') > fitness(s_k)$
17. Remove solution s_k from the k th nest
18. Put solution s' into the k th nest
19. **end if**
20. **end while**
21. Select the best solution among all nests

Table 4.4. Fitness of the different solutions found by CS using different number of nests

Number of nests	Number of facilities					
	8	18	48	50	72	127
2	72.02	206.45	719.79	646.06	1196.24	2013.67
5	72.02	206.97	722.99	645.30	1195.67	2008.25
10	72.02	201.75	718.47	646.54	1196.38	2019.30
15	72.08	202.76	720.30	645.75	1185.14	2003.64
20	69.20	204.60	719.27	644.79	1188.04	2017.85
25	71.77	195.47	724.88	646.29	1197.16	2008.37

4.5. Comparison

To analyze the results obtained with the three previously presented algorithms, GA, SA and CS, we also computed an additional simple method which solves individual LA where each facility decides its service by simply selecting the service that maximizes its demand without taking into account the other facilities. In order to compare the four algorithms we applied them to ten different sized clusters obtained with Algorithm 3.2 (see Chapter 3). The results are presented in Table 4.5. There we can see that the individual LA is much faster than the other algorithms since it does not have to perform several iterations before giving the final solution. However, the results obtained with this algorithm are much worse than the results provided by SA, GA and CS. This fact justifies the maximization of the global demand instead of the individual maximization of the demand.

On the other hand, if we compare the three presented methods (GA, SA and CS) we can say that SA is able to provide the best results. Most of its results have the greatest fitness (global weighted demand), the highest number of allocated customers and the lowest number of empty facilities (we consider empty facilities those with an occupation lower than 4%). Although, the three algorithms achieve results with small differences in terms of fitness and number of allocated customers, SA provides results with a very low number of empty facilities which are not able to achieve GA and SA, especially when the number of facilities grows, so when the problem grows in complexity.

If we focus in the computational cost of each algorithm, we can state that GA is the most costing algorithm in terms of time and memory resources since it is the algorithm that spends more time to get a final solution and also because it has to work with 100 solutions approximately in each iteration, thus it spends a lot of memory resources. This is the reason for which we do not have GA results for the case with 1495 facilities (last row of Table 4.5). On the other hand, both SA and CS, just generates one solution in each iteration, what speeds up them in comparison to GA and they keep in memory two (SA) and eleven (CS) solutions what decrease the memory resources they need; for this reason SA is the most efficient algorithm.

In order to combine the best algorithm (SA) and the fastest one (individual LA) we decided to analyze the SA performance when it is initialized using the solution found with individual LA. The elapsed time of the combination of both algorithms is not greater than the SA elapsed time (since the time needed by the individual LA algorithm is negligible). However, it is able to find slightly better results (see Table 4.6).

Appendix B exposes the images of the clusters in order to show how they are.

Table 4.5. LA results. Best results are in bold face.

Number of facilities	Fitness				% of allocated customers				% of facil. with occupation < 4%				Elapsed time (s)			
	Individual	GA	SA	CS	Individual	GA	SA	CS	Individual	GA	SA	CS	Individual	GA	SA	CS
8	81.39	109.56	108.27	107.30	56.73	79.30	78.13	78.13	0.00	4.29	0.00	0.00	0.000	0.467	0.129	0.136
18	170.38	279.91	281.86	278.35	51.39	94.16	95.72	95.82	0.00	1.11	0.00	1.11	0.001	3.103	0.662	0.702
42	438.26	707.69	723.27	713.74	56.94	99.88	99.83	99.55	0.00	12.61	0.00	0.48	0.009	17.164	4.140	4.083
46	427.11	681.92	706.08	696.38	55.50	98.17	99.68	99.54	2.17	13.06	2.61	3.48	0.009	11.741	2.440	2.878
48	479.4	824.50	838.18	832.65	53.85	99.50	99.58	99.58	0.00	4.58	0.00	0.00	0.011	22.155	5.660	6.146
50	484.39	754.45	776.96	768.91	57.10	97.58	97.97	97.94	2.00	12.40	0.00	1.20	0.004	16.409	4.067	4.323
72	622.92	1057.11	1079.42	1074.73	54.89	98.89	98.97	98.89	0.00	4.58	3.06	3.06	0.021	34.486	11.088	11.553
127	1389.85	2374.754	2421.44	2404.44	55.58	100.00	100.00	100.00	0.79	14.80	0.16	1.57	0.028	159.720	50.617	48.039
313	3019.05	5144.42	5258.10	5238.18	55.75	99.58	99.75	99.74	0.32	21.15	0.58	3.07	0.136	712.152	293.865	288.316
1495	14660.55	-	25826.85	25762.79	55.91	-	99.97	99.99	0.07	-	0.54	3.28	3.571	-	5285.298	4934.568

Table 4.6. LA results using SA initialized with individual LA solution and SA initialized randomly

Number of facilities	Fitness		% of allocated customers		% of facilities. with occupation < 4%		Elapsed time (s)	
	Individual LA & SA	SA	Individual LA & SA	SA	Individual LA & SA	SA	Individual LA & SA	SA
18	259.89	257.42	97.52	97.38	0.00	0.00	0.612	0.591
42	707.57	704.87	99.88	99.88	2.38	2.38	4.579	4.546
72	1229.38	1222.67	98.95	98.92	1.39	1.39	14.549	14.747
127	2242.62	2234.65	100	100.00	0.79	2.76	49.237	49.893
313	5077.28	5068.38	99.83	99.77	1.28	2.08	299.298	299.293
1495	26259.56	26229.77	99.97	99.99	0.54	1.27	6079.012	5976.626

CHAPTER 5. Conclusions

This Master's Thesis has opened some problems that should be addressed in a future work.

The aim of this research is to solve the immobile LA problem specifying it to work out the best match each bar should broadcast in order to maximize the global weighted demand. As said in Chapter 2, the LA problem under study is a k^n combinatorial problem where k is the number of possible matches and n the number of bars. Due to the complexity of the problem we first decided to simplify the problem converting it to several smaller problems using a clustering analysis of the bars to divide the initial problem. Thus, the objective of this research is to study and solve a concrete LA problem using clustering analysis to convert the initial problem into sub-problems and then solving these sub-problems applying optimization techniques. Therefore in Chapter 2 we provide an exhaustive state of the art about three main topics: clustering methods, LA problem and optimization techniques.

In Chapter 3, we explored some clustering techniques in order to find the clustering method that finds the most appropriate division of the problem. Therefore, this research has contributed with the use of clustering techniques in order to simplify the LA problem. We tested with k-means, Lloyd's algorithm, region growing, hierarchical clustering, genetic clustering and affinity propagation. We found that k-means provide the best results, though region growing is the fastest algorithm and the only available one for databases with more than 10000 elements approximately.

Next, in Chapter 4, we analyzed some optimization techniques in order to find a near optimal solution of the LA problem for the clusters of bars. Concretely we analyzed three metaheuristic techniques (GA, SA and CS) and we compared their performances in order to find the fittest method to such problem, which has been SA. We also contributed with this Master's Thesis presenting a new neighborhood function that speeds up the convergence of the SA and CS algorithms. It consists in assigning, to each bar, an exponential probability function that depends on its occupancy. We also made a comparison with other possible neighborhood functions in order to give an evidence of its good performance.

CHAPTER 6. Future work

In Chapter 3 we had the aim to find an appropriate clustering method to divide the initial problem into sub-problems; however we found two methods that are the fittest in some particular cases, i.e. k-means seems to be the most appropriate algorithm when the dataset of bars are not larger than 10000 elements approximately, but that region growing is the only (between the analyzed) feasible method for very large datasets. So it remains hanging to work out a clustering method that provides a good division of the problem. It would be interesting to also explore other partition techniques besides clustering.

In Chapter 4 we solved the sub-LA problems using three metaheuristic techniques (GA, SA and CS). We discarded the use of other incomplete optimization techniques due to the features of the problem. For example we discarded the use of linear programming because the search space is not convex, or we also discarded the use of techniques such as hill climbing or gradient based techniques because there are, in the problem search space, several local optimums. We also excluded the use of complete optimization techniques due to the dimensions of the search space. Therefore, it would be interesting to work out a complete optimization method to solve this problem feasibly. It would be also interesting to use some acceleration techniques for the metaheuristic methods analyzed here in order to speed up them.

Regarding the distance measures between customers and bars, we have used the Euclidean distance, but in practice it would be interesting to complete this research using the true distances between customers and bars according to the streets and roads and adding some features of the bars like the type of food, the favorite team of the bar, etc. and the customers preferences. Additionally, a system that tells the bars the match they should broadcast and informs the customers where they can watch their desired match should rank or weight every bar according to a measure of confidence.

Moreover, in this research we supposed that the known position of the customers corresponds to their position just before the match, what does not have to be true. So it would be needed to develop an estimator of the customers' position just before the match using their known positions and historical data.

Finally, in this research we solved the LA problem of each cluster separately. However, sometimes, at the clusters' borders there may be some customers close to bars of different

clusters. Thus, a possible future work is to consider some kind of permeability of the clusters' frontiers with the customers. This may be solved using techniques for DCSP.

Appendix A. Numeric distances

Here we expose the definition of some useful distances that can be used for the different topics of this Master's Thesis.

- Aitchison, [139]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^D \left(\log\left(\frac{x_{ik}}{g(\mathbf{x}_i)}\right) - \log\left(\frac{x_{jk}}{g(\mathbf{x}_j)}\right) \right)^2}, \quad g(\mathbf{x}) = \left(\prod_{k=1}^D x_k \right)^{1/D}$$

- Angular, [140]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \arccos \left(\frac{\sum_{k=1}^D \sqrt{x_{ik} x_{jk}}}{\sqrt{\sum_{p=1}^D x_{ip}^2 \cdot \sum_{q=1}^D x_{jq}^2}} \right)$$

- Bhattacharyya (arccos), [141]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \arccos \left(\sum_{k=1}^D \sqrt{x_{ik} x_{jk}} \right)$$

- Bhattacharyya (log), [141]

$$d(\mathbf{x}_i, \mathbf{x}_j) = -\log \left(\sum_{k=1}^D \sqrt{x_{ik} x_{jk}} \right)$$

- Bray Curtis, [142]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^D |x_{ik} - x_{jk}|}{\sum_{l=1}^D (x_{il} - x_{jl})}$$

- City-Block, [143]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D |x_{ik} - x_{jk}|$$

- Euclidean, [144]

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^D (x_{ik} - x_{jk})^2}$$

- Hellinger, [145]

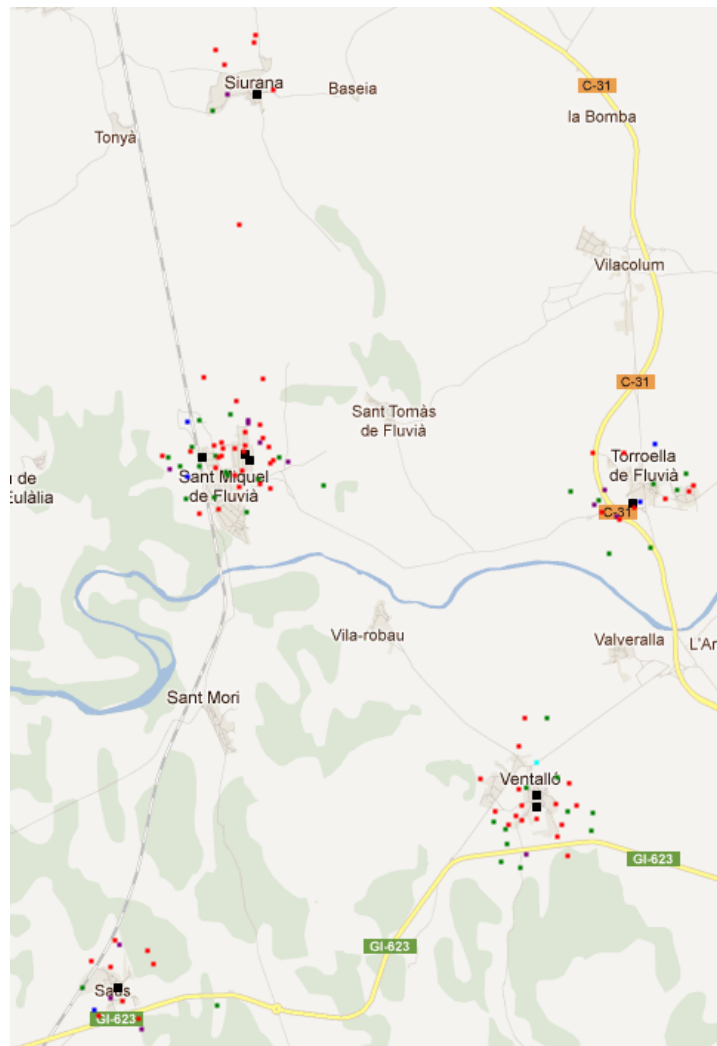
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^D (\sqrt{x_{ik}} - \sqrt{x_{jk}})^2}$$

- Mahalanobis, [146]

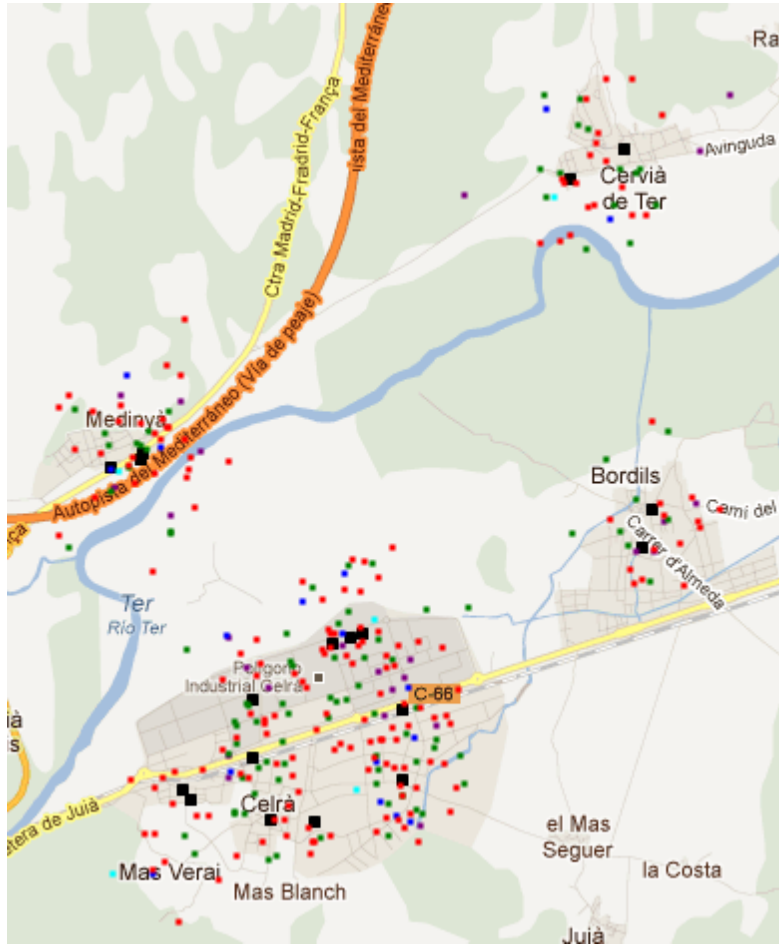
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{C}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}, \quad \mathbf{C} = \mathbf{x}_i \mathbf{x}_j^T$$

Appendix B. Cluster images

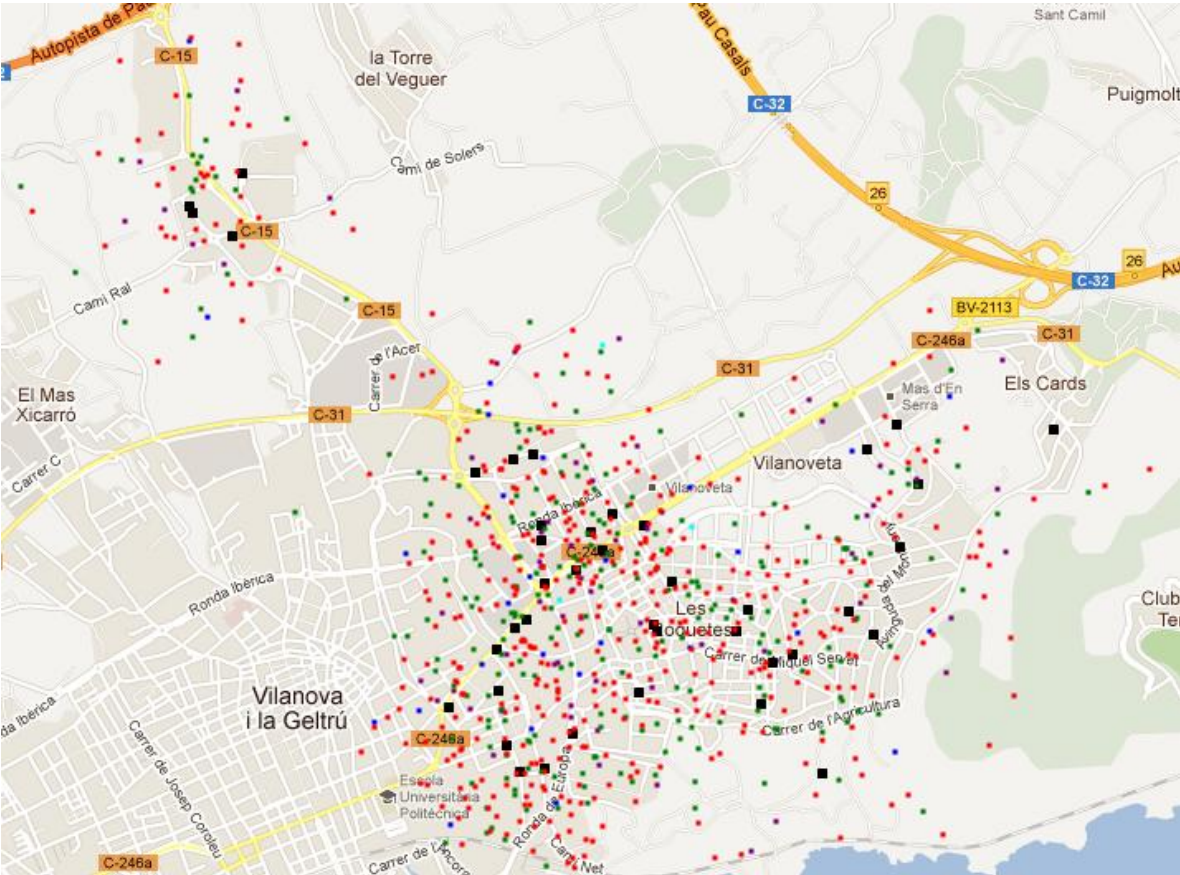
Here we show the images of the clusters used to analyze the performance of the LA algorithms used.



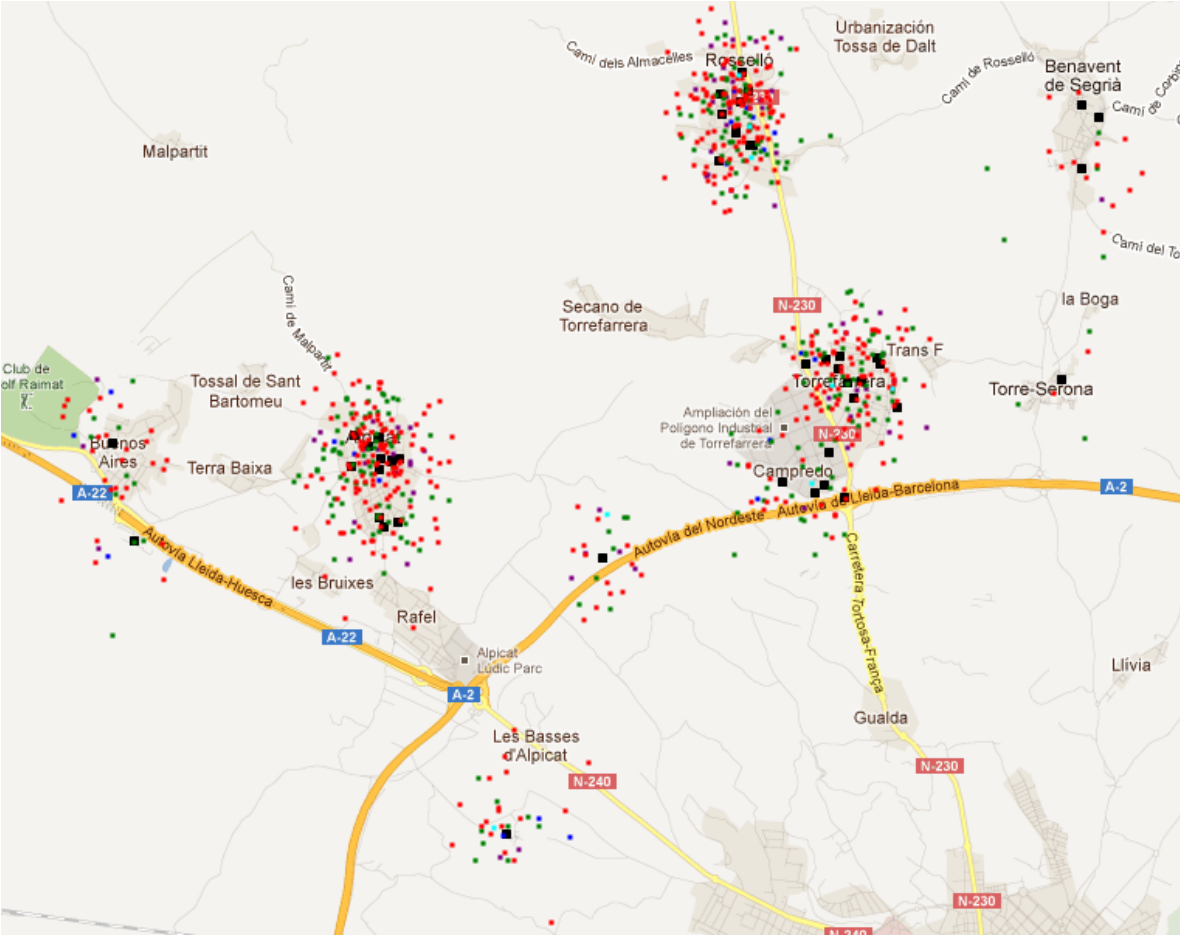
a) Cluster with 8 facilities



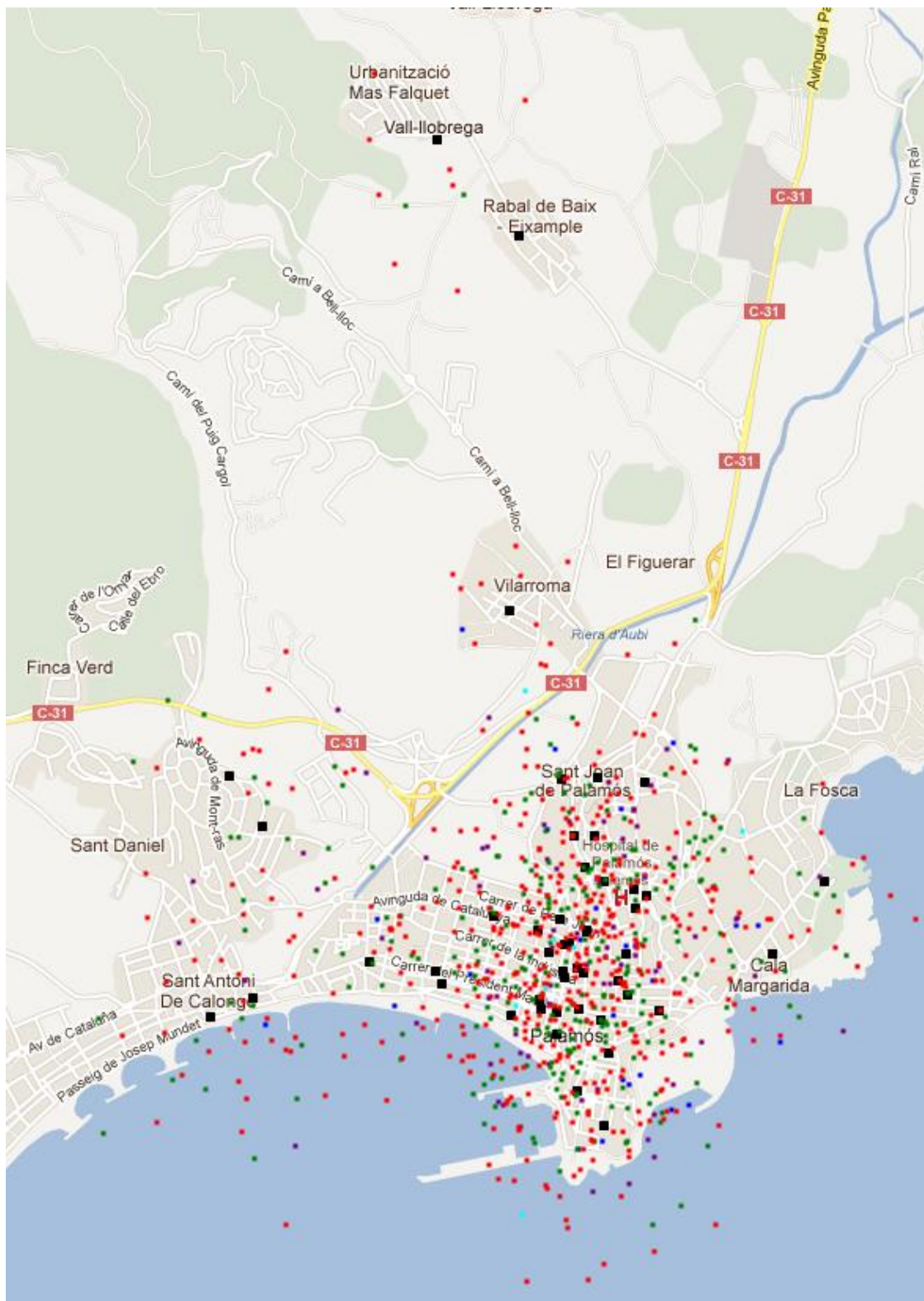
b) Cluster with 18 facilities



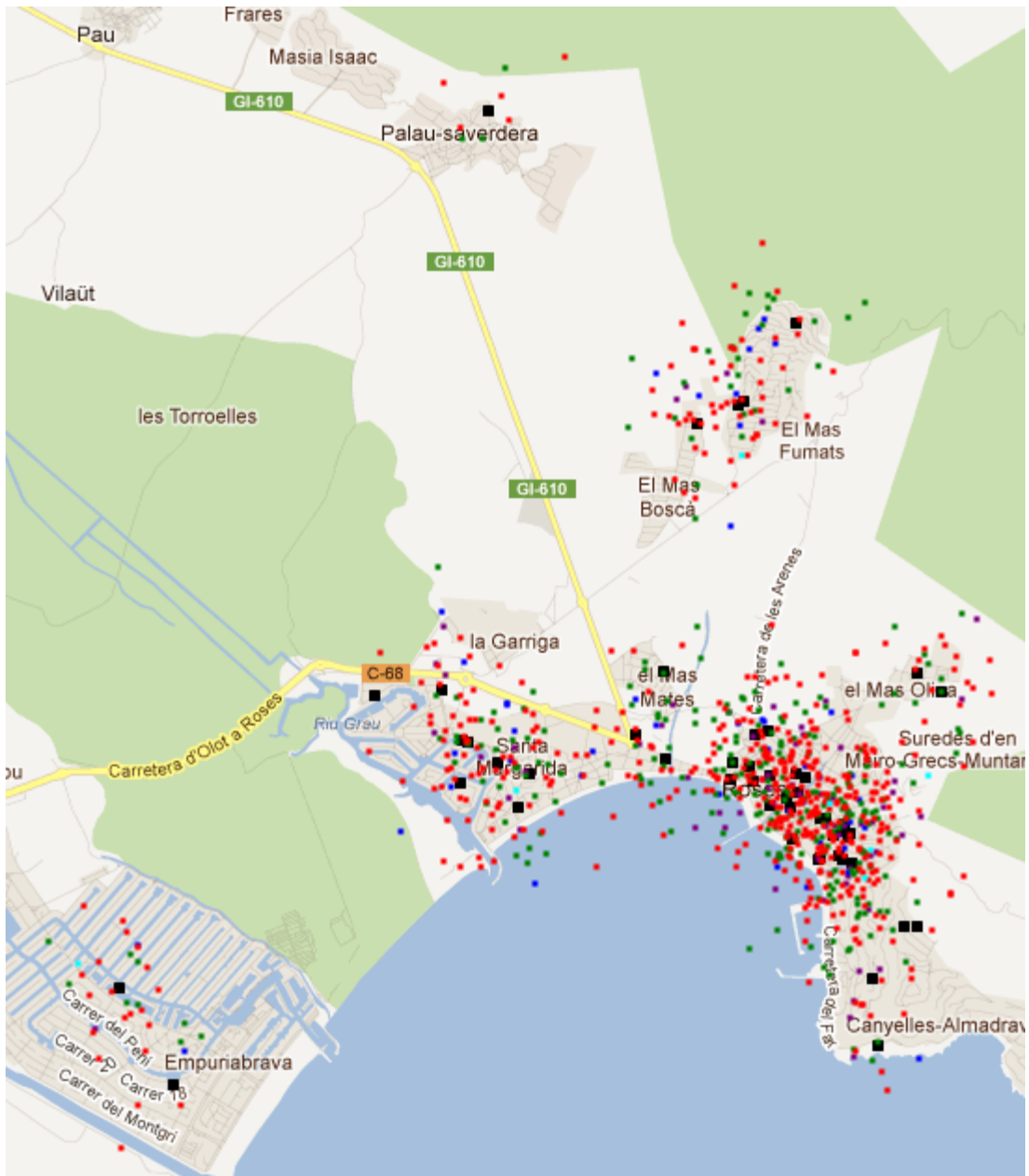
c) Cluster with 42 facilities



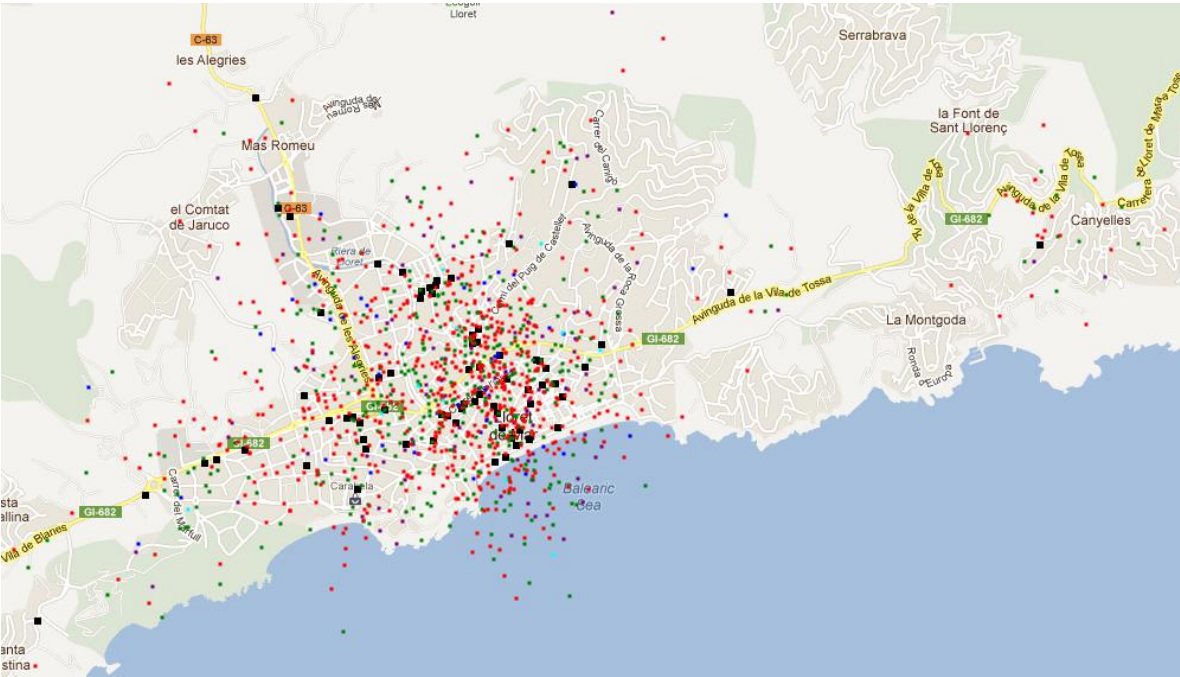
d) Cluster with 46 facilities



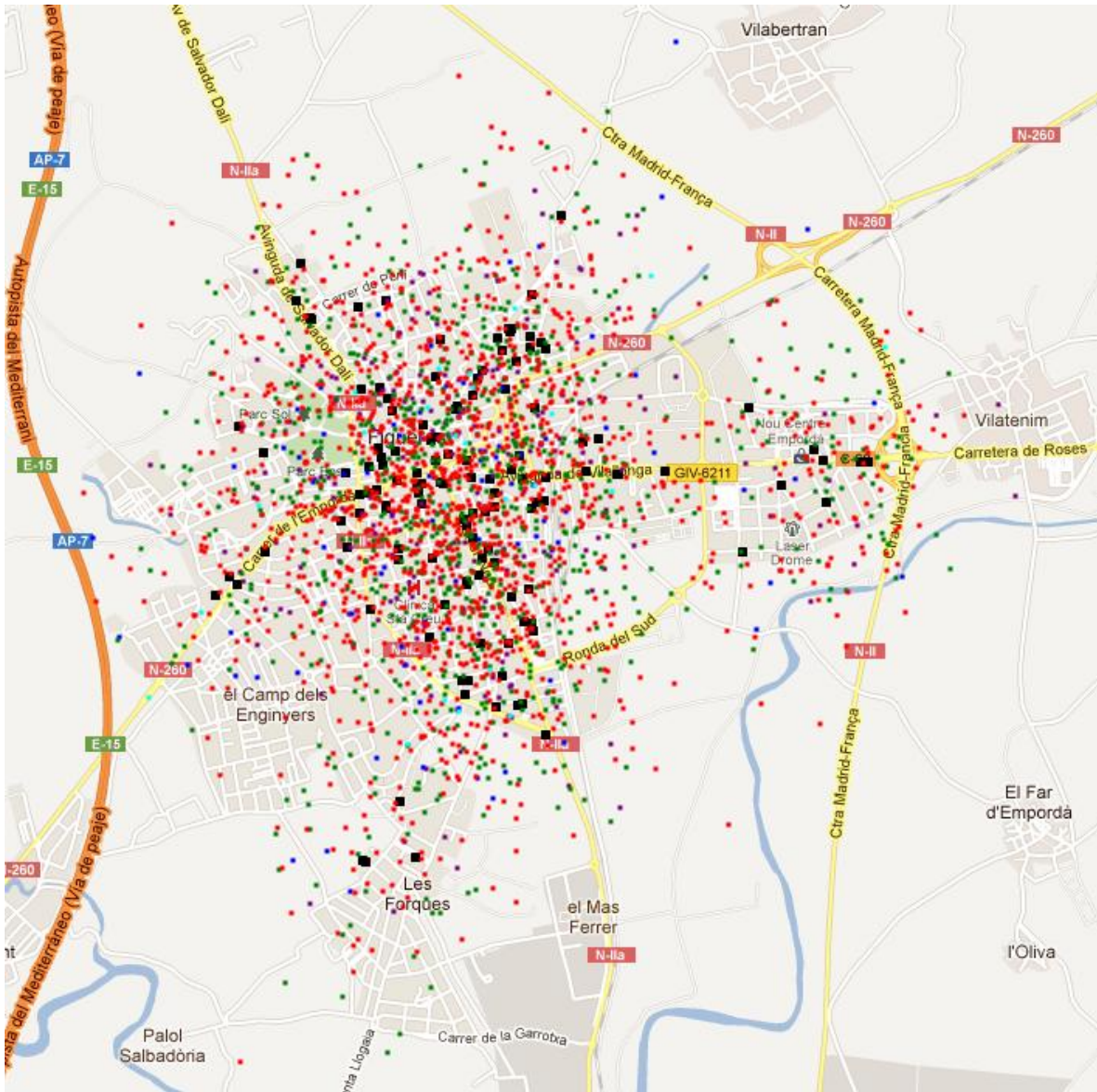
e) Cluster with 48 facilities



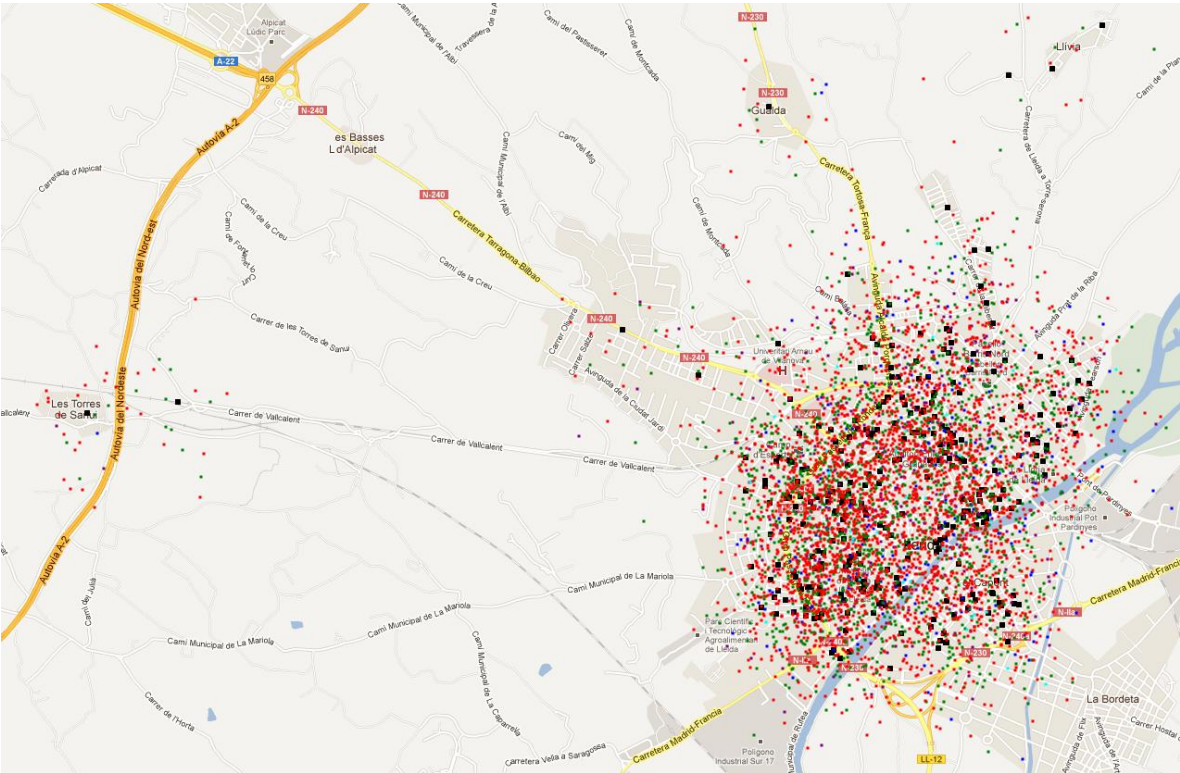
f) Cluster with 50 facilities



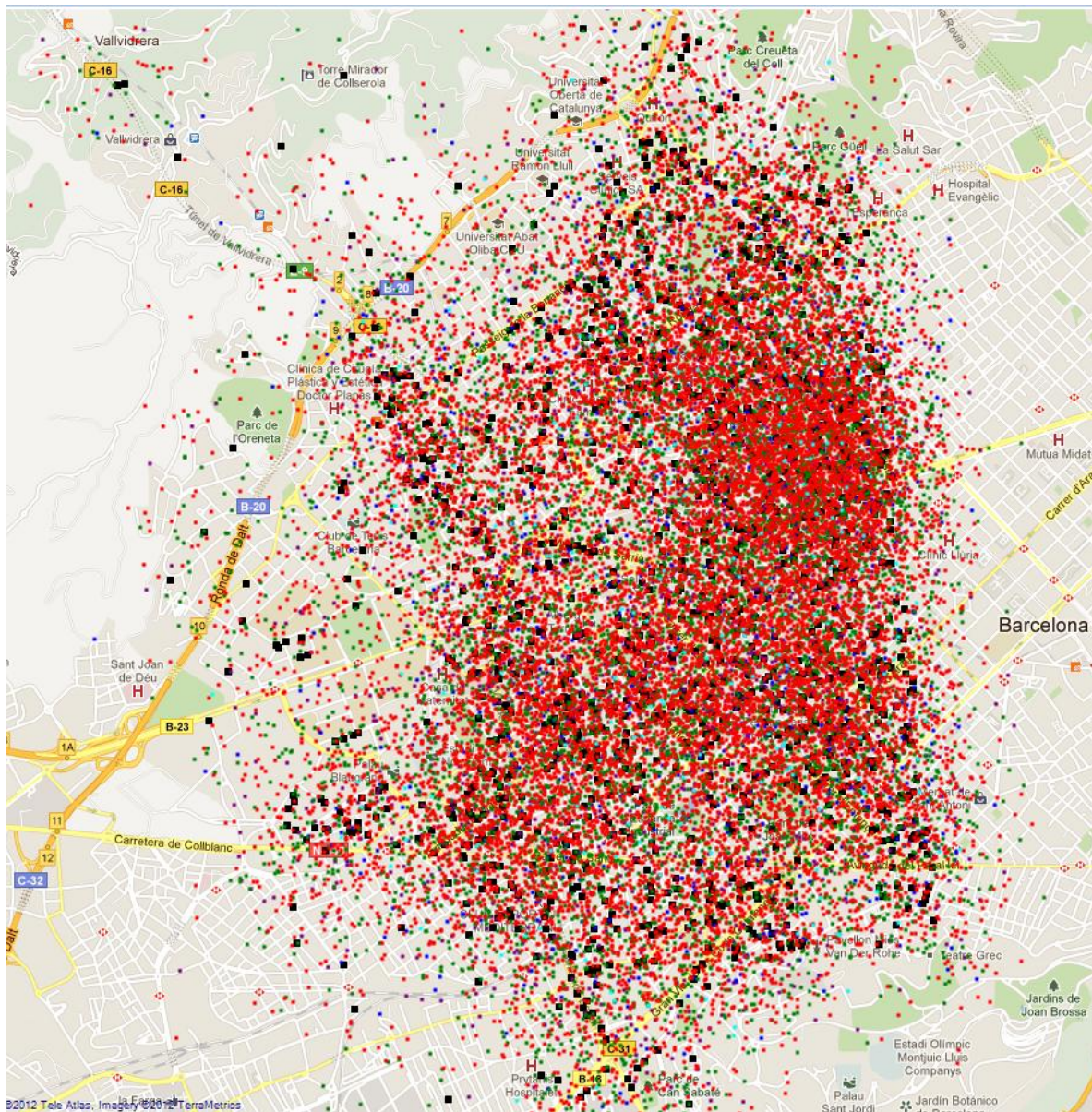
g) Cluster with 72 facilities



h) Cluser with 127 facilities



i) Cluster with 313 facilities



j) Cluster with 1495 facilities

Bibliography

- [1] R. R. Sokal and P. H. A. Sneath, "Principles of numerical taxonomy," *Systematic Biology*, vol. 13, no. 106-108, pp. 1-4, 1963.
- [2] I. C. Lerman, *Les bases de la classification automatique*, Gauthier-Villars, 1970.
- [3] N. Jardine and R. Sibson, *Mathematical taxonomy*, London: John Wiley, 1971.
- [4] M. R. Anderberg, "Cluster analysis for applications," DTIC Document, 1973.
- [5] E. J. Bijnen, *Cluster analysis: survey and evaluation techniques*, 1973.
- [6] H. Bock, *Automatische klassifikation*, 1974.
- [7] W. Sodeur, *Empirische Verfahren zur Klassifikation*, 1974.
- [8] F. Vogel, *Probleme und Verfahren der numerischen Klassifikation*, Vandenhoeck & Ruprecht, 1975.
- [9] H. Späth, *Cluster-Analyse-Algorithmen*, Munich: München und Wien, 1975.
- [10] J. A. Hartigan, *Clustering algorithms*, John Willey & Sons, 1975.

- [11] N. Mantel, "The detection of disease clustering and a generalized regression approach," *Cancer research*, vol. 27, no. 2, p. 209, 1967.

- [12] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241-254, 1967.

- [13] E. H. Ruspini, "A new approach to clustering," *Information and control*, vol. 15, no. 1, pp. 22-32, 1969.

- [14] P. F. Jonsson, T. Cavana, D. Zicha and P. A. Bates, "Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metasis," *BMC Bioinformatics*, vol. 7, no. 1, p. 2, 2006.

- [15] M. C. Horner-Devine and J. M. Bohannon, "Phylogenetic clustering and overdispersion in bacterial communities," *Ecology*, vol. 87, pp. 100-108, 2006.

- [16] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu and T.-J. Chen, "Fuzzy c-means clustering with spatial information for image segmentation," *Computerized Medical Imaging and Graphics*, vol. 30, no. 1, pp. 9-15, January 2006.

- [17] G. Punj and D. W. Stewart, "Cluster analysis in marketing research: review and suggestions for application," *Journal of Marketing*, vol. 20, pp. 134-148, 1983.

- [18] M. S. Handcock, A. E. Raftery and J. M. Tantrum, "Model-based clustering for social networks," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 170, no. 2, pp. 301-354, 2007.

- [19] R. Huth, C. Beck, A. Philipp, M. Demuzere, Z. Ustrnul, M. Cahynov, J. Kysely and O. E. Tveito, "Classification of atmospheric circulation patterns," *Annals of the New York Academy of Sciences*, vol. 1146, pp. 105-152, 2008.

- [20] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 881-892, jul 2002.

- [21] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 3, pp. 433-439, jun 1999.
- [22] L. Jing, M. Ng and J. Huang, "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 8, pp. 1026-1041, aug. 2007.
- [23] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, 2003, pp. 1713 - 1723.
- [24] G. Karypis, E.-H. Han and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68-75, aug 1999.
- [25] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Norwell, MA, USA: Kluwer Academic Publishers, 1981.
- [26] R. Krishnapuram and J. Keller, "A possibilistic approach to clustering," *Fuzzy Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 98-110, may 1993.
- [27] R. Krishnapuram and J. Keller, "The possibilistic C-means algorithm: insights and recommendations," *Fuzzy Systems, IEEE Transactions on*, vol. 4, no. 3, pp. 385-393, aug 1996.
- [28] S. P. Lloyd, "Least square quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129-137, 1982.
- [29] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027-1035, 2007.
- [30] R. Mojena, "Hierarchical grouping methods and stopping rules: an evaluation," *The computer Journal*, vol. 20, no. 4, pp. 359-363, 1977.
- [31] D. Dembélé and P. Kastner, "Fuzzy C-means method for clustering microarray data,"

Bioinformatics, vol. 19, pp. 973-980, 2003.

- [32] S. Y. Kim and T. M. Choi, "Fuzzy types clustering for microarray data," *International Journal of Computational Intelligence*, vol. 2, pp. 12-15, 2006.
- [33] H.-L. Shieh, Y.-K. Yang and C.-N. Lee, "A robust fuzzy clustering approach and its application to Principal Component Analysis," *Intelligent Automation and Soft Computing*, vol. 16, no. 1, p. 1, 2010.
- [34] V. Tseng and C.-P. Kao, "A Novel Similarity-Based Fuzzy Clustering Algorithm by Integrating PCM and Mountain Method," *Fuzzy Systems, IEEE Transactions on*, vol. 15, no. 6, pp. 1188-1196, dec. 2007.
- [35] K.-L. and Du, "Clustering: A neural network approach," *Neural Networks*, vol. 23, no. 1, pp. 89-107, 2010.
- [36] R. Sheikh, M. Raghuwanshi and A. Jaiswal, "Genetic Algorithm Based Clustering: A Survey," in *Emerging Trends in Engineering and Technology, 2008. ICETET '08. First International Conference on*, 2008, pp. 314-319.
- [37] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-WesleyEditors, Ed., Addison-Wesley, 1989, p. 432.
- [38] P. Kudova, "Clustering Genetic Algorithm," in *Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on*, 2007, pp. 138-142.
- [39] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 226-231.
- [40] J. A. Hartigan, "Direct clustering of a data matrix," *J. Amer. Statist. Assoc.*, vol. 67, no. 337, pp. 123-132, 1972.
- [41] R. Bekkerman, R. El-Yaniv and A. McCallum, "Multi-way distributional clustering via pairwise interactions," *Proceedings of 22nd International Conference on Machine Learning*, pp. 41-48,

2005.

- [42] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, p. 972, 2007.
- [43] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-Theory and Methods*, vol. 3, no. 1, pp. 1-27, 1974.
- [44] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100-108, 1979.
- [45] W. J. Krzanowski and Y. T. Lai, "A criterion for determining the number of groups in a data set using sum-of-squares clustering," *Biometrics*, pp. 23-34, 1988.
- [46] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, Wiley Online Library, 1990.
- [47] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224-227, 1979.
- [48] A. Weber, *Über den Standort der Industrien*, JCB Mohr, 1909.
- [49] W. Isard, *Location and Space Economy*, Cambridge: Cambridge Mass: The MIT Press, 1956.
- [50] L. Cooper, "Location-allocation problems," *Operations Research*, pp. 331-343, 1963.
- [51] S. Openshaw and P. Streadman, "On the geography of a worst case nuclear attack on the population of Britain," *Political Geography Quarterly*, vol. 1, no. 3, pp. 263-278, 1982.
- [52] X. Li, Z. Liu and X. Zhang, "Applying Genetic Algorithm and Hilbert Curve to Capacitated Location Allocation of Facilities," in *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on*, vol. 1, 2009, pp. 378-383.

- [53] L. Li, Q. YanYou and L. Wei, "Overview of optimization algorithms in facility allocation problems," in *2010 Sixth International Conference on Natural Computation (ICNC)*, vol. 3, 2010, pp. 1143-1147.
- [54] S. Sasaki, A. J. Comber, H. Suzuki and C. Brunsdon, "Using genetic algorithms to optimise current and future health planning - the example of ambulance locations," *International Journal of Health Geographics*, vol. 9, no. 1, p. 4, 2010.
- [55] A. J. Comber, S. Sasaki, H. Suzuki and C. Brunsdon, "A modified grouping genetic algorithm to select ambulance site locations," *International Journal of Geographical Information Science*, vol. 25, no. 5, pp. 807-823, 2011.
- [56] R. L. Church, "Location modelling and GIS," *Geographical Information Systems*, vol. 1, pp. 293-303, 1999.
- [57] S. H. Pasandideh and S. T. Niaki, "Genetic application in facility location problem with random demand within queuing framework," *Journal of Intelligent Manufacturing*, pp. 1-9, 2010.
- [58] K.-H. Hsieh and F.-C. Tien, "Self-organizing feature maps for solving location-allocation problems with rectilinear distances," *Computers and Operations Research*, vol. 31, no. 7, pp. 1017-1031, 2004.
- [59] X. Li and A. G.-O. Yeh, "Integration of genetic algorithms and GIS for optimal location search," *International Journal of Geographical Information Science*, vol. 19, no. 5, pp. 581-601, 2005.
- [60] S. Kongsomsaksakul, A. Chen and C. Yang, "Shelter location-allocation model for flood evacuation planning," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 6, pp. 4237-4252, 2005.
- [61] P. X. Li and F. Y. Qiang, "Solving Competitive Facilities Location Problem with the Clonal Selection Algorithm," in *Management Science and Engineering, 2006. ICMSE '06. 2006 International Conference on*, 2006, pp. 413-417.

- [62] R. Aboolian, O. Berman and D. Krass, "Competitive facility location and design problem," *European Journal of Operational Research*, vol. 182, no. 1, pp. 40-62, 2007.
- [63] L. Wang-sheng, Z. Jing-fa and L. Mao-qing, "A genetic algorithm approach to location-allocation problem in Urban Garbage Logistics System," in *IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on*, 2008, pp. 71-75.
- [64] M. Neema and A. Ohgai, "Multi-objective location modeling of urban parks and open spaces: Continuous optimization," *Computers, Environment and Urban Systems*, vol. 34, no. 5, pp. 359-376, 2010.
- [65] M. Wen and K. Iwamura, "Facility location-allocation problem in random fuzzy environment: Using (α, β) -cost minimization model under the Hurewicz criterion," *Computers & Mathematics with Applications*, vol. 55, no. 4, pp. 704-713, 2008.
- [66] Z. Le, J. J. Hua and L. W. Hua, "Research on Distribution Center's Location-Allocation Problem with Flexible Allocation Strategy," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 4408-4410.
- [67] M. Wen and R. Kang, "Some optimal models for facility location-allocation problem with random fuzzy demands," *Applied Soft Computing*, vol. 11, no. 1, pp. 1202-1207, 2011.
- [68] J. Redondo, J. Fernández, I. García and P. Ortigosa, "Sensitivity analysis of a continuous multifacility competitive location and design problem," *TOP*, vol. 17, pp. 347-365, 2009.
- [69] V. Yassenovskiy and J. Hodgson, "Hierarchical Location-Allocation with Spatial Choice Interaction Modeling," *Annals of the Association of American Geographers*, vol. 97, no. 3, pp. 496-511, 2007.
- [70] R. Courant and H. Robbins, *What is Mathematics?*, Oxford University, 1996.
- [71] C. Kimberling's, "Encyclopedia of Triangle Centers," 2003.
- [72] E. Zeleny, Lagrange Points, Wolfram Demonstrations Project.

- [73] A. Björck, Numerical methods for least squares problems, Philadelphia: SIAM, 1996.
- [74] M. Avriel, Nonlinear Programming: Analysis and Methods, Dover Publishing, 2003.
- [75] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.
- [76] R. A. Howard, Dynamic probabilistic systems, New York: John Wiley & Sons, 1971.
- [77] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the 16th annual ACM symposium on Theory of Computing*, 1984, pp. 302-311.
- [78] W. Karush, "Minima of functions of several variables with inequalities as side conditions," University of Chicago, Department of Mathematics, Chicago, 1939.
- [79] S. Russell and P. Norvig, Artificial Intelligence. A modern approach, New Jersey: Pearson Education, 2010.
- [80] R. Tarjan, "Depth-first search and linear graph algorithms," in *12th Annual Symposium on Switching and Automata Theory*, 1971.
- [81] T. Min, Y. Ren-mu and S. Yan, "A depth-first search algorithm based implementation approach of spanning tree in power system," *Power System Technology*, vol. 34, no. 2, pp. 120-124, 2010.
- [82] B. Vukojevic, N. Goel, K. Kalaichevan, A. Nayak and I. Stojmenovic, "Power-aware depth-first search based georouting in ad-hoc and sensor wireless networks," in *9th IFIP International Conference on Mobile Wireless Communications Networks*, 2007.
- [83] F. Ricca, W. Faber and N. Leone, "A backjumping technique for disjunctive logic programming," *AI Communications*, vol. 19, no. 2, pp. 155-172, 2006.
- [84] X. Chen and P. Van Beek, "Conflict-directed backjumping revisited," Ithaca, 2011.

- [85] E. A. Galburt, S. W. Grill, A. Wiedman, L. Lubkowska, J. Choy, E. Nogales, M. Kashlev and C. Bustamante, "Backtracking determines the force sensitivity of RNAP II in a factor-dependent manner," *Nature*, vol. 446, no. 7137, pp. 820-823, 2007.
- [86] M. Voliotis, N. Choen, C. Molina-Paris and T. B. Liverpool, "Fluctuations, pauses and backtracking in DNA transcription," *Biophysical journal*, vol. 94, no. 2, pp. 334-348, 2008.
- [87] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497-520, 1960.
- [88] C. H. Lampert, M. B. Blaschko and T. Hofman, "Efficient subwindow search: a branch and bound framework for object localization," *IEEE Transactions on Pattern Analysis and machine intelligence*, vol. 31, no. 12, pp. 2129-2142, 2009.
- [89] Y. Utsumi, Y. Matsumoto and Y. Iwai, "An efficient branch and bound method for face recognition," in *Signal and Image Processing Applications*, IEEE, 2009, pp. 156-161.
- [90] A. D'Adriano, D. Pacciarelli and M. Pranzo, "A branch and bound algorithm for scheduling trains in a railway network," *European Journal of Operational Research*, vol. 183, no. 2, pp. 643-657, 2007.
- [91] N. Nilsson, "Some growth and ramification properties of certain integrals on algebraic manifolds," *Arkiv för Matematik*, vol. 5, no. 5, pp. 463-476, 1964.
- [92] C. C. Green and B. Raphael, "Research on intelligent question-answering system," DTIC Document, 1967.
- [93] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, 1968.
- [94] L. Xiang and D. Gong, "A comparative study of A* algorithms for search and rescue in perfect maze," in *Electric Information and Control Engineering*, 2011, pp. 24-27.

- [95] L. V. Kantorovich, "The mathematical method of production planning and organization," *Management Science*, vol. 6, pp. 363-422, 1939.
- [96] G. B. Dantzig, "Computer for solving bombing," US Patent 2.421.745, 1947.
- [97] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203-4215, 2005.
- [98] K. Yang, X. Wang and J. Feldman, "A new linear programming approach to decoding linear block codes," *IEEE Transactions on Information Theory*, vol. 54, no. 3, pp. 1061-1072, 2008.
- [99] X. Zhang and P. H. Siegel, "Adaptative cut generation for improved linear programming decoding of binary linear codes," in *Information Theory Proceedings*, 2011, pp. 1638-1642.
- [100] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289-1306, 2006.
- [101] K. Chakrabarty, S. S. Iyengar, Q. Hairong and C. Eungchun, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448-1453, 2002.
- [102] D. Muramatsu, "Online signature verification using Hill climbing method," in *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2008, pp. 133-138.
- [103] J. A. Snyman, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient Based Algorithms*, Springer Publishing, 2005.
- [104] M. Á. Lagunas, "Capítulo V: Sistemas adaptativos," in *Procesado de señal*, 2007.
- [105] F. W. Glover, "Tabu Search - Part 1," *ORSA Journal on Computing*, vol. 1, no. 2, pp. 190-206, 1989.
- [106] F. W. Glover, "Tabu Search - Part 2," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4-32,

1990.

- [107] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
- [108] V. Cerný, "Thermodinamical approach to the traveling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, pp. 41-51, 1985.
- [109] C. C. Queirolo, L. Silva, O. P. Bellon and M. P. Segundo, "3D face recognition using simulated annealing and the surface interpenetration measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 206-219, 2010.
- [110] O. Ekren and B. Y. Ekren, "Size optimization of PV/wind hybrid energy system with battery storage using simulated annealing," *Applied Energy*, vol. 87, no. 2, pp. 592-598, 2010.
- [111] W. Li, Y. W. Chen, J. F. Li and F. Yao, "Approach to remotely sensed data processing task scheduling problem based on fast simulated annealing," *Systems Engineering and Electronics*, vol. 33, no. 2, pp. 334-338, 2011.
- [112] N. A. Barricelli, "Esempi numerici di processi di evoluzione," in *Methodos*, 1954, pp. 45-68.
- [113] N. A. Barricelli, "Symbiogenetic evolution processes realized by artificial methods," in *Methodos*, 1957, pp. 143-182.
- [114] A. Fraser, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Aust. J. Biol. Sci.*, vol. 10, pp. 484-491, 1957.
- [115] A. Fraser, *Computer Models in Genetics*, New York: McGraw-Hill, 1970.
- [116] I. Rechenberg, *Evolutionsstrategie*, Stuttgart: Holzmann-Frobbog, 1973.
- [117] H.-P. Schwefel, "Numerische Optimierung von Computer-Modellen," PhD Thesis, 1974.

- [118] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*, Stuttgart: Birkhäuser, 1977.
- [119] J. Holland, *Adaptation in Natural and Artificial Systems*, Michigan, 1975.
- [120] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, Springer, 2004.
- [121] A. Coloni, M. Dorigo and V. Maniezzo, "Distributed optimization by ant colonies," in *Première Conférence Européenne Sur la Vie Artificielle*, Paris, 1991.
- [122] M. Dorigo, "Optimization, learning and natural algorithms," PhD Thesis, Politecnico di Milano, Italy, 1992.
- [123] S. Okdem and D. Karaboga, "Routing in wireless sensor networks using ant colony optimization router chip," *Sensors*, vol. 9, no. 2, pp. 909-921, 2009.
- [124] K. L. Huang and C. J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1030-1046, 2008.
- [125] M. H. Aghdam, N. Ghasem-Aghaee and M. E. Basiri, "Text feature selection using ant colony optimization," *Expert systems with applications*, vol. 63, no. 3, pp. 6843-6853, 2009.
- [126] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.
- [127] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *Proceedings of IEEE International conference on Evolutionary Computation*, pp. 69-73, 1998.
- [128] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," *Proceedings of Evolutionary Programming*, vol. 7, pp. 591-600, 1998.

- [129] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 84-88, 2000.
- [130] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.
- [131] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letter*, vol. 85, no. 6, pp. 317-325, 2003.
- [132] M. R. AlRashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 913-918, 2009.
- [133] S. Pandey, L. Wu, S. M. Guru and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environment," in *24th International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, 2010, pp. 400-407.
- [134] K. Tamura and Y. Keiichiro, "Spiral optimization," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2011, pp. 1759-1764.
- [135] K. Tamura and K. Yasuda, "Primary study of spiral dynamics inspired optimization," *IEEE Transactions on Electrical and Electronic Engineering*, vol. 6, no. 1, pp. 98-100, 2011.
- [136] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *World Congress on Nature & Biologically Inspired Computing*, IEEE Publications, 2009, pp. 210-214.
- [137] X.-S.-. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.
- [138] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recognition*, vol. 35, no. 6, pp. 1197-1208, 2002.

- [139] J. Aitchison, "On criteria for measures of compositional difference," *Mathematical Geology*, vol. 24, no. 4, pp. 365-379, 1992.
- [140] E. W. Weisstein, "Angular distance," in *MathWorld*.
- [141] A. Bhattacharyya, "On a measure of divergence between two multinomial populations," *The Indian Journal of Statistics (1933-1960)*, vol. 7, no. 4, pp. 401-406, 1946.
- [142] J. R. Bray and J. T. Curtis, "An ordination of the upland forest communities of southern Wisconsin," *Ecological monographs*, vol. 27, no. 4, pp. 325-349, 1957.
- [143] E. F. Krause, *Taxicab Geometry*, Dover, 1987.
- [144] E. Deza and M. M. Deza, "Encyclopedia of Distances," Springer, p. 94.
- [145] Y. Grace and L. C. Lucien, *Asymptotics in Statistics: Some Basic Concepts*, Berlin: Springer, 2000.
- [146] P. C. Mahalanobis, "Mahalanobis distance," *Proceedings National of Science of India*, vol. 49, no. 2, pp. 234-256, 1936.
- [147] R. Hertwig and T. Pachur, *Heuristics*, New York: Oxford University Press, 2011.
- [148] I. H. Osman and J. P. Kelly, *Meta-Heuristics: An Overview*, Dordrecht: Kluwer Academic Publishers, 1996.

