TOWARDS A PARALLEL SYSTEM FOR DEMOGRAPHIC ZONIFICATION BASED ON COMPLEX NETWORKS

Alberto Ochoa[1], Beatríz Bernábe[2], Omar Ochoa[3]

[1]Institute of Cybernetics, Mathematics and Physics. ochoa@icmf.inf.cu
[2]UNAM, Systems Department & BUAP School of Computer Science. bety@cs.buap.mx
[3]Institute of Cybernetics, Mathematics and Physics. omar@icmf.inf.cu

ABSTRACT
This paper presents a novel method for the zone design problem that is based on a popular technique in the field of complex networks research: betweenness centrality. We use a parallel version of a community detection algorithm that outputs the graph partitioning that maximizes the so-called modularity metric. The method is put at the centre of an effort to build an open source interactive high performance computing platform to assist researchers working with population data.

Keywords: parallel algorithms, zone design problem, betweenness centrality, clustering, interactive high performance computing.

RESUMEN
Este artículo presenta un método novedoso para el problema de diseño de zonas, que se basa en una técnica muy popular en el campo de las investigaciones de redes complejas: la intermediación de aristas. Se utiliza una versión paralela de un algoritmo de detección de comunidades que retorna el particionamiento del grafo que maximiza la llamada métrica de modularidad. El método se pone en el centro de un esfuerzo en pos de la construcción de un sistema de código abierto para computación interactiva de alto rendimiento que asista a los investigadores que trabajan con datos demográficos.

Palabras clave: algoritmos paralelos, diseño territorial, intermediación central, agrupamiento, computación paralela de alto rendimiento.

1. Introduction

The ultimate goal of the present work is the creation of a system to assist researchers that investigate population data. The social and economic impact of dealing with this kind of information can be assessed from the following sources [1, 2, 3].

Specifically, we address the problem of demographic zonification, which is understood here as the problem of clustering geographical units (GU) according to some measure of demographic similarity. Each GU is a small geographical area for which the following is known: latitude, longitude, radius and a vector of demographic features (statistics) computed over the population living in the area. A cluster in the intended solution is expected to be compact, which means that each GU is geographically close to a large fraction of the GUs in the same cluster. Besides, the cluster should be homogeneous as to the utilized similarity measure, meaning each GU has rather high similarity values with many members of its cluster. In other words, the clustering can be seen as a two-objective optimization problem or as a single objective one with the restriction of getting compact solutions. In this paper, we take the later approach.

Theoretically speaking, solving the above defined clustering problem is challenging. In terms of computational complexity, the zone design problem has been shown to be NP-Complete [4]. One of the reasons why this problem is especially difficult is the size of the solution space. The dimension of most real world problems makes unfeasible any exact solution. Thus, heuristic techniques seem to be the best way available to produce solutions in a reasonable computational time. Unfortunately, the heuristics used so far have also a high computational cost [5, 6] and often do not provide good solutions. It is worth noting that approaches based on classical statistics alone cannot solve the problem either [7]. We believe it is necessary to apply a combination of heuristics and statistical methods to obtain good results. In this paper, we explore new ideas and develop better algorithms, including parallelism as an option.

To make things even more challenging, we would like to create an open source tool for assisting in situ people that may have no or limited knowledge of parallelism and lack the necessary software and equipment. These people work interactively with population data, which means that they may change the set of features that describe the GUs, the geographical regions of interest or the metrics. Besides, they need statistical tools for analysing their results. The system must fulfil all these requirements. This paper is a first step towards the construction of such a tool.

The method we have developed in the paper is motivated by results in complex network research [8, 9, 10, 11, 12]. Basically, what we do is to map the problem to an unweighted graph and then apply a community detection algorithm -based on edge betweenness centrality- that produces the desired partitioning of the GUs. However, the map from the distance and similarity matrices to the unweighted graph may not be so straightforward as far as a lot of weight information has to be codified in the graph topology.

The outline of the paper is as follows. Section 2 is devoted to a detailed exposition of the proposed method. The method is applied to the clustering of geographic units according to given demographic information, but with the restriction of geographic closeness. In Section 3, a hint is given about how to apply the method to the clustering of the demographic variables. Section 4 presents a short overview of the first steps given in the design and building of an open source interactive high performance computing platform to assist researchers working with population data. Section 5 shows some numerical results that illustrate the method in action. Finally, the conclusions are given.

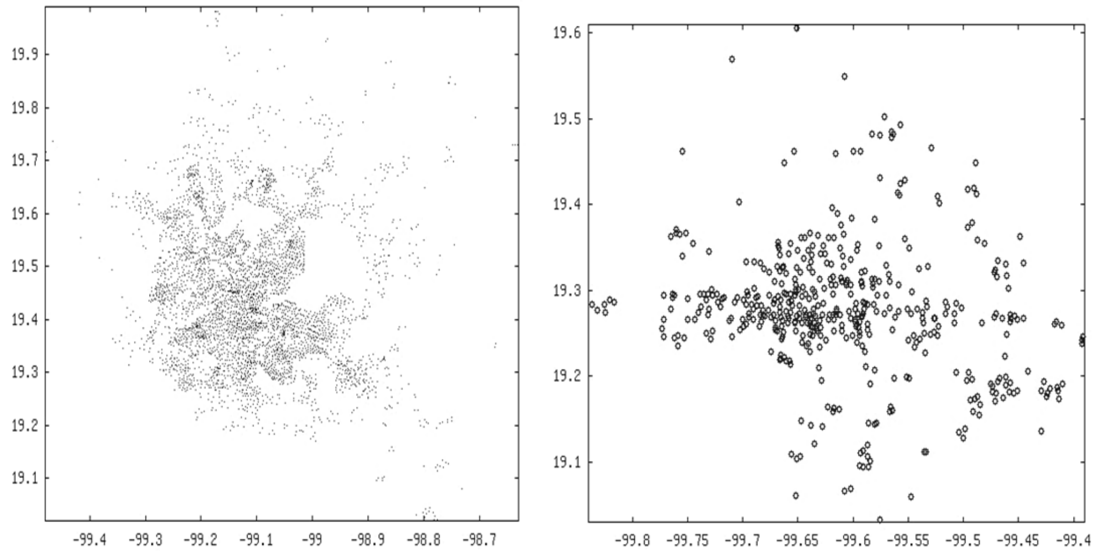


Figure 1. Geographical distribution (latitude vs. longitude) of 4292 and 466 GUs of the metropolitan areas of the valleys of Mexico (left) and Toluca, respectively. A smaller marker size has been used in the first case.

2. Spatially constrained demographic clustering

Throughout the paper, we shall use a simplified model of the zone design problem. Our goal is just to show that a method based on network partitioning is suitable for the problem. According to the model, the geographic units are assumed to be points.

As it was said, the problem of demographic zonification is understood in this paper as the problem of clustering geographical units according to demographic similarities. Figure 1 shows the distribution of a set of such units for two areas of Mexico: the metropolitan areas of the valleys of Toluca and Mexico. We shall use both areas in our experiments, but the smaller region (Toluca) has been chosen as the main case study throughout the paper.

The main ideas of our algorithm, described next, are based on observations and conclusions drawn from research in complex networks. Indeed, we transform the problem of clustering the GUs according to two measures: demographic similarity and Euclidean distance, into the problem of community detection in an unweighted graph. As an alternative, we could also work with a weighted graph, which would give a more straightforward representation of our heavily weighted problem. However, as far as dealing with the unweighted case is less expensive from the computational point of view we accepted the challenge of mapping the weight information into the structure of an unweighted graph.

One important decision to be made before the algorithm starts is the selection of a metric to measure the demographic similarity between two GUs.

```
Algorithm 1 Our algorithm
Assume as given a similarity measure and an input matrix where rows codify the geographic units (GUs).

Phase I   Selection of variables, geographic units and the space of processing.
Phase II  Computation of the set of closest neighbors of each GU, according to
          the distance information.
          Step 1  Computation of the Delaunay triangulation of the set of GUs.
                  This creates the triangulated graph, $G_t$, where each node
                  represents a GU.
          Step 2  Edge and node elimination in $G_t$.
          Step 3  To every node, $i$ in $G_t$, assign the set of its closest neighbors
          $Neigh_d(i)$.
Phase III Computation of the similarity graph, $G_s$.
          Step 1  For every node, $i$, compute its pairwise similarity with every
                  member of $Neigh_d(i)$.
          Step 2  For each node, $i$, construct the set, $Neigh_s(i)$ (s stand for
                  similarity), with the $k$ most similar neighbors taken among the
                  members of $Neigh_d(i)$.
          Step 3  Add the edge $i \sim j$ to $G_s$ if $j \in Neigh_s(i)$ and $i \in Neigh_d(j)$.
Phase IV Parallel computation of the edge betweenness centrality of every edge in
          $G_s$.
Phase V  Computation of the partitioning of $G_s$ that maximize a given scoring
          metric after eliminating some of the edges with the highest betweenness values.
```

There exists a wide choice of metrics for this purpose, among them the Euclidean, cosine and correlation functions are the most popular. In this paper, we use the Euclidean and cosine functions for clustering the GUs and the correlation function for clustering the demographic variables.
In what follows, the details of the method, which is outlined in Algorithm 1, are discussed.

2.1 Selection of variables, GUs and the space

Let us assume that an input data matrix is given, where each row contains information about one of the GUs, whereas the columns represent the demographic and geographic variables. There are many reasons that make convenient to work with a subset of the variables and/or a subset of the available GUs.

The selection of one or another group of variables for processing gives different perspectives on the data, which often is useful in interactive exploratory data analysis. Besides, the reduction of the numbers of variables decreases the arithmetic complexity of the computation of the similarity function. It is worth noting, that in our case this gain is not so important because, as will be shown below, our method makes a linear amount of pairwise similarity computations with respect to the number of GUs.

Based on the above said, we can conclude that any automatic variable selection method can be plugged-in in the Phase I of the algorithm. Another kind of techniques that can be utilized in this stage is the transformation of the space where the data reside, which includes principal component analysis and other projection algorithms. It is expected that the transformation will lead to the computation of better similarity values as far as they will be more robust in the presence of noise.
From the perspective of their geographical positions, a subset of the available GUs may be chosen when we want to concentrate ourselves on the study of a particular area of the given region or when the computation of the whole region is expensive according to the available equipment. We shall present examples of both situations in the section dedicated to the experiments. Another very important perspective on the same issue arises when solving the clustering problem for a subset of GUs resulting from a complex logical database query with respect to the demographic variables.

In summary, Phase I is responsible for producing an input matrix that better fulfils the requirements of the clustering task at a given moment.

2.2 Creating the unweighted similarity graph

Recall that in our problem, a cluster in the desired solution is expected to be compact, which means that each GU is geographically close to a large fraction of the GUs in the same cluster. At the same time, the cluster should be homogeneous as to the utilized similarity measure, meaning each GU has rather high similarity values with many members of its cluster. In other words, the clustering can be seen as a two-objective optimization problem or as a single objective one with the restriction of getting compact solutions. We take the later approach and as a first step create a graph based on the distance information.

Without loss of generality, let us assume that the second and third columns of the input matrix contain the latitude and longitude of each GU (these columns were not processed in Phase I). It is worth noting that we are not considering the radius, which means that all GUs are assumed to be geometrically equal with centres at the given coordinates. In some cases, this assumption could be too strong and somehow misleading. However, it is adequate for the purposes of this paper as explained below.

The Delaunay triangulation (DT) is a sparse graph structure that codifies proximity relationships in a point set. Thus, the first thing we do in Phase II is the construction of the unweighted graph Gt-a DT of the set of GUs centres. The DT is connected to the Voronoi diagram of the centres, which is a partition of the space into areas, one for each GU centre, so that the area for a GU centre consists of that region of the space that is closer to it than to any other GU centre. An edge in the DT connects two GU centres if and only if the Voronoi areas containing them share a common boundary. Hence, edges of the DT capture spatial proximity. At this point, it is worth noting that for those cases where the actual area of each GU is included completely inside of the Voronoi cell of its centre, the DT seems to be adequate to model the distance restriction of our problem. Regarding computational efficiency, many O (n log n) time and O (n) space algorithms exist for computing the DT of a planar point set.

In Step 2, we eliminate a certain amount of nodes and edges to get a further reduction of the graph (the first was performed with the input data in Phase I). Optionally, we eliminate GUs in regions of high density to reduce the cost of the algorithm. This is accomplished as follows: We compute a DT of the whole set and then select the GUs belonging to triangles which areas are not below the Q area quantile. A new triangulation is performed with the chosen points.

Despite the fact that edges of the DT capture spatial proximity, there are some edges that might be far apart, for example the edges between the GUs in the convex hull. Therefore, we remove all edges larger than the largest one found in a set that contains for each GU the smallest among its attached edges. We term Gt the connected graph obtained after this procedure.

As the final step of Phase II, we compute the set of closest neighbors of the i-th GU, Neighd (i) (where the d indicates the distance measure). Having in mind the way Gt was constructed, it is reasonable to define as neighbors all GUs that can be reached in at most L steps from the given one. In fact, in all the experiments of this paper, we use L = 2.

At this point, Phase III of the algorithm starts. The idea is simple, we formulate a k-nearest neighbors problem with respect to the similarity function. Fortunately, we only need to compute the similarity between each GU and its neighbors according to Neighd (i) (Step 1). For each GU, i, construct the set, Neighs (i) (s stands for similarity), with the k most similar neighbors taken among the members of Neighd (i) (Step 2).

```
Algorithm 2 Community Detection by Girvan and Newman

While the network is not empty,

  Step 1  Recalculate the betweenness score of every edge.

  Step 2  Remove the edge with the highest score.
```

In the final Step 3 the unweighted graph, Gs, is constructed. The following holds:

edge i~j $\in G_s$ if only if (j $\in$ Neigh$_s$ (i)) & (i $\in$Neigh$_s$ (j)).

It is our fundamental claim that in the topology of the unweighted graph Gs there is enough information about our complex clustering problem. If we assume that, and we do, it is reasonable to accept as clusters all the connected sub-graphs with high connection density. Therefore, what we need at this point is a method to detect these communities of nodes. This is a common problem in complex networks research.

2.3 Edge betweenness centrality

The shortest path edge betweenness centrality counts the number of shortest paths that run along an edge in a graph. The highest values of betweenness are likely to be associated with bridges-edges -those that separate regions of densely connected vertexes. Thus, by removing the bridges the communities can be detected. Unfortunately, when there are several bridges between two regions, they compete for the best scores, difficulting an accurate evaluation of the measure. This problem has been solved with Algorithm 2, hereafter called the GN algorithm because it was developed by Girvan and Newman [8, 10].

The fastest known version of the GN algorithm for unweighted, undirected graphs (m edges and n vertexes) has worst case time complexity $O(m^2n)$, whereas for sparse graphs $O(n^3)$. The high computational cost of the GN clustering algorithm has been an obstacle to its use on relatively large graphs. In fact, the computational cost of the algorithm is already prohibitive when the input is a graph with a few thousand edges. However, in [12] a parallel implementation of the algorithm was reported. As far as it allows users to analyse larger networks on a distributed cluster of computers, we decide to evaluate it in our problem.

This is accomplished in Phase IV of the algorithm. The output at this stage is the order of removal of the edges, which implicitly defines a hierarchical tree. Each node of the tree is a subset of GUs.

2.4 Partitioning the similarity graph

The last phase of our algorithm is responsible for deciding where to cut the hierarchical tree to determine the clusters. There are different ways to accomplish this task. Here, we use a recently introduced quality measure for graph clustering called modularity [10]. A high modularity indicates that there are more intra-cluster and less inter-cluster edges than would be expected by chance. Note that this differs from partitioning a graph minimizing the number of inter-cluster edges.

Assuming that aij represents the fraction of all edges that link the vertices in cluster i to the vertices in cluster j, the modularity is then defined as

$$0 \leq Q = \sum_{i=1}^{K} \left(a_{ii} - \left(\sum_{j=1}^{K} a_{ij} \right)^2 \right) \leq 1$$

The higher is the value, the stronger is the clustering structure in the network. For a random decomposition Q approaches 0.

The version of the algorithm reported in this paper outputs the clustering with maximum modularity and the ordered list of edge removals. Optionally, the user can take the list of edges and by gradually removing them from the graph he or she can create a merge matrix or dendrogram from which different clustering can be explored.

3. Clustering the demographic variables

The very same complex networks approach can be utilized to cluster the demographic variables before and after running Algorithm 1. In the first case, the aim is the selection of variables and GUs in Phase I, whereas in the second it is helpful for labelling the clusters of GUs.

Here, we use 1 - corr(X, Y) as a measure of similarity between variables X and Y. Note that corr stands for correlation (Pearson or Spearman) and no missing values are allowed to ensure corr(X, Y) ≤ 1. Strong correlated variables are assumed to belong to the same cluster unless the corresponding edge has a high betweenness value, which is a sufficient condition for separating variables.

The unweighted similarity graph is constructed by attaching to each variable its k-most similar variables and adding the edge ‹X, Y›, if the relation holds in both directions. Once the graph is built, Phases IV and V of Algorithm 1 can be executed.

4. A parallel system for zonification

The term interactive high performance computing (IHPC) has to do with a system's ability to provide access to parallel hardware to run programs written in a very high language. The most common example is to accelerate Matlab by running the programs in a computer cluster. Most IHPC systems come with several mathematical libraries. We are interested in statistical and complex networks libraries. ViPoC (Virtual Interactive Portable Cluster [13, 14]) is one experimental IHPC that fulfils our requirements.

We are building an experimental open source parallel tool for demographic zonification including a priori and posterior statistical analysis. We pursue five aims: 1) allow rapid prototyping via high level languages like R and Matlab; 2) large collection of statistical and machine learning tools; 3) cheap parallel infrastructure; 4) smooth integration with existing Windows Matlab technology; and, 5) web interface with the system. ViPoC provides all these things.

4.1 A short overview of VIPOC

ViPoC is a flexible portable cluster because it can be used in a laptop, in a small office network or in a big organization managing a large cluster. It controls two sub-networks, one is formed by dedicated nodes behind a switch and the other may have a mixture of dedicated and no dedicated nodes -some of them may be virtual-located at the intranet of the organization. Therefore, it is easy to build a low budget cluster.
ViPoC supports the R and Octave languages, among many others. The later is a clone of Matlab, whereas the former has many statistical and machine learning functions ready to use. This is an important characteristic for the system we want to build: a large choice of statistical methods can be used in the a priori and posterior analysis of the zone design problem.

ViPoC comes with three so-called IHPC servers. They provide socket access to Octave, R and the system itself. For example, from Matlab running in the user's desktop computer it is possible to issue the following commands:

```
callViPoC (`command’, vipocIP)                        (1)
callViPoCOctave (`command’, ListArgs, vipocIP)        (2)
callViPoCR (‘command’, ListArgs, vipocIP              (3)
```

where vipocIP is the ip-address of the ViPoC server and ListArgs is a list of arguments that depends on the concrete command. For example, the call

```
callViPoC(‘mpirun -np 4 pgn AGEBs.pebc > getclusters.m’, vipocIP)
```

executes parallel program pgn in 4 processors. This program is a modification of the code reported in [12]. It takes file AGEBs.pebc (the list of edges in Gs) as input and writes to the standard output the results (in form of a Matlab program) which is redirected to file getclusters.m.

In ViPoC, the working directory in the user's desktop computer can be mapped to the user's working directory in the cluster (via the Samba protocol). Therefore, the Matlab function getclusters.m can be called from the user's Matlab program obtaining, in this way, the computed clusters.

Let us assume that matrix AGEBdata is a Matlab variable in the user's desktop computer that contains the geographic and demographic information of each GU. That matrix is usually the input to Algorithm 1, but sometimes we may want to do some pre-processing. Figure 2 presents a fragment of code that computes the principal components of the matrix, calling the R server of ViPoC (see [13] for syntactic details).

This fragment of code corresponds to Phase I of Algorithm 1. Notice that the same approach can be utilized after the run of our algorithm to do some statistical post-processing using R.

At this point, it should be clear for the reader that our approach to the construction of the system consists in using different languages and systems to maximize reusability. In the current version, we proceeded as follows: Phase I is implemented following the ideas outlined in Figure 2. In step 1 of Phase II, a very efficient C++ program obtained from the internet was used. The remainder of Phase II and Phase III were programmed in Matlab (Octave). This part of the program can be run in the user's desktop computer in Matlab or Octave, or completely in the cluster in Octave. The parallel computation of Phase IV is accomplished as explained above; it returns the best clustering according to the modularity measure. Other reasonable good clustering can be obtained, again via the ViPoC R server, using the complex network package igraph [15]. The final post-processing is accomplished in the same way. It is worth noting that R also has the capacity to run parallel code.

Before we conclude this short presentation of the use of ViPoC as a computational platform for our system, we stress the following fact once again: We are interested in providing a system that researchers and practitioners can use in their available hardware, from a single laptop to a large cluster. In the later case, it is also advisable to have a web interface for remote access. ViPoC provides this option.

5. Experimental results

The aim of this section is simply to illustrate the possibilities of our method. Let us take the data for the metropolitan area of the valley of Toluca. We studied 466 GUs described with a vector of 181 demographic variables according to the XII National Population Census [16].

In the first experiment, we use all GUs and variables (without any pre-processing) and the Euclidean distance as the demographic similarity measure. The distance graph was built as explained above using two levels of the triangulated graph. For the construction of the similarity graph, we took the 15 most similar GUs, which produced a sparse graph with 2937 edges. The algorithm found the nine clusters shown in Figure 3 (bottom).

```
if Preprocessing
 callViPoCR ( ' open ' );
 callViPoCR ( ' assign ', 'X ' , AGEBdata( : , 4: end ) );
 callViPoCR ( ' eval ', ...
               sprintf ( ' pc <- prcomp(X, scale=TRUE, ...
               center=TRUE, tol =%1.3f)', pcTol));
pcAGEBdata = callViPoCR ( 'get_matrix ', ' pc$x ' );
AGEBdata = [ AGEBdata(:, 1 : 3 ) pcAGEBdata ];
callViPoCR (' close ');
end
```

Figure 2. Fragment of Matlab (or Octave) code that uses the ViPoC R server to compute the principal component of matrix AGEBdata (each row represents one GU). Although we have removed the ip-address for the sake of space, this code can be executed from the user's desktop computer.

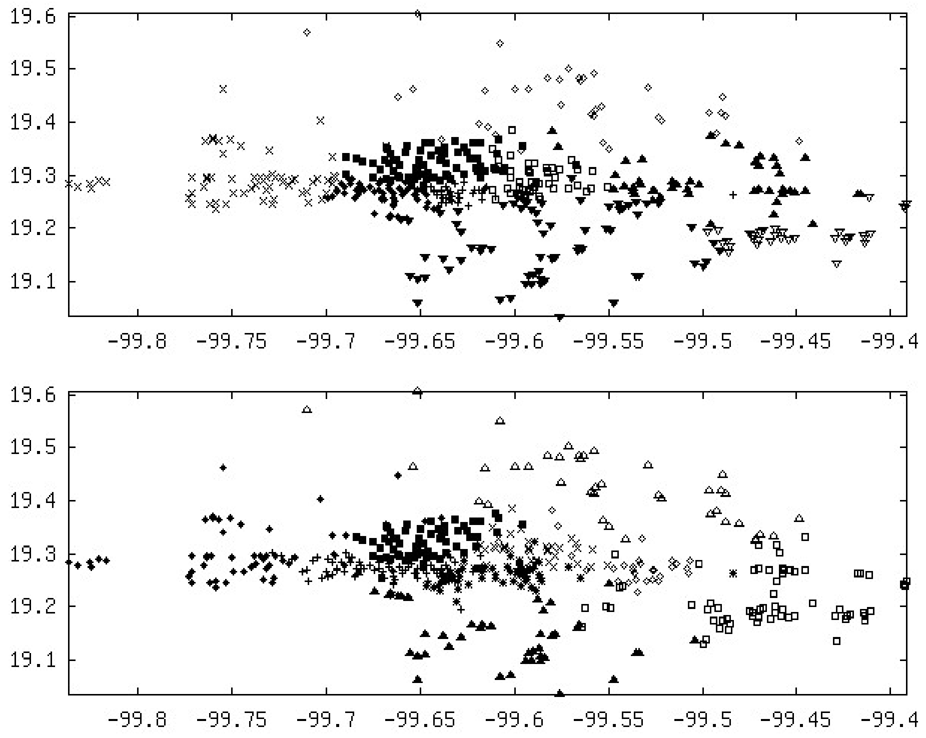


Figure 3. Clustering results for Toluca using the whole set of 181 demographic variables. The top figure shows a clustering where only geographic distances were used whereas for the bottom figure Euclidean distances between the vectors of demographic variables were used as well.

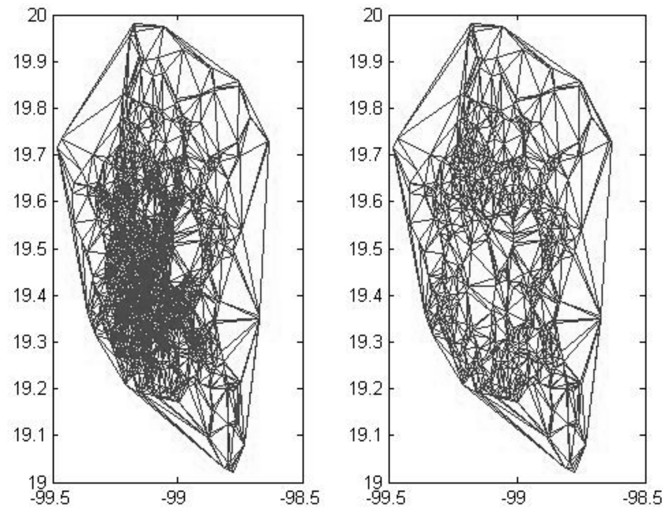


Figure 4. Reducing the number of GUs for more efficient yet approximate clustering. (left) Delaunay triangulation of 4292 GUs in the Metropolitan Valley of Mexico. (right) Triangulation of the GUs belonging to triangles, whose areas are not below the 0.90 area quantile. The number of GUs drops to 836 Gus.

For comparison purposes, in Figure 3 (top), we show the clustering obtained using exclusively the geographic distances. To accomplish this, we applied the betweenness method to the distances' graph, which has 4289 edges. The important thing to notice is that the bottom partitioning cannot be trivially obtained from the top one.

For example, the cluster represented by squares at the bottom of the figure contains GUs from three different clusters of the top clustering (three different triangles). Note that only one of these clusters is completely included in the clusters of squares. Thus, the idea of clustering with one measure and then with the other does not seem to be a good one. Our method achieves a reasonable compromise between the two measures, because the first one is utilized to constraint the solution space in such a way that the optimization of the second does not violate the requirements of the first one.

When dealing with larger problems like the one presented in Figure 1 (left), the issue of obtaining a response in a reasonable computational time arises.

In an interactive scenario, like the one we are trying to construct, there are different heuristics that can be applied. Just to give an example, we have shown in Figure 4 a simple method that consists in reducing the number of GUs for more efficient yet approximate clustering. This is accomplished by doing a two step triangulation. Only the GUs belonging to triangles which areas are not below the 0.90 area quantile in the first triangulation are considered in the second triangulation. We recall that the triangulation algorithm has complexity *O* (*n* log *n*). In the example, the number of considered GUs drops from 4292 to 835.

A second strategy, which of course can be combined with heuristics, is the use of parallel or distributed programs. The size of the manageable problem will depend on the available computational power.

Nevertheless, we can always restrict ourselves to the analysis of small areas as far as we are working in an interactive scenario. For example, in Figure 5 (top) the rectangular area that has been chosen for processing is shown in red.

We use ViPoC to solve this problem in parallel. Table 1 shows the results of a small experiment. We took four clients with P-IV 2.4 GHz processors supporting hyper-threading technology in a 800 MHz bus and 512 MB of RAM. The clients were connected through a Fast-Ethernet network switch to a laptop where ViPoC was running. The table shows the time to complete Phases IV and V of our algorithm.

The data to be clustered is the rectangular area mentioned above: 875 Gus and 5573 edges. Notice that an approximate time reduction from 14 to 8 minutes is achieved when 5 processors were used.

This experiment has shown an unexpected and interesting issue. Notice that starting from six processors, the computational time increases. This

Processors	1	2	3	4	5	6	7
Time (secs)	854.14	689.75	524.19	498.61	476.72	486.17	525.06

Table 1. Time (seconds) for different number of processors. Parallel computation of the best partitioning of a similarity graph with 875 vertices and 5573 edges, constructed for the rectangular area shown in Figure 5.

is in contrast with the published in [12] linear speedup of up to 32 processors. We have found that this algorithm does not scale well with fast-Ethernet networks. We assume that the reported results were obtained with much faster network technologies. The problem is that the implementation of the betweenness algorithm (due to the known recalculation step [8]) needs a certain amount of inter-processor communication that becomes prohibited already for six processors in slow networks. This is a great obstacle for the purposes of our project because we are specially targeting low-cost hardware. Fortunately, in [11] a method (differential betweenness) whose parallel implementation does not require such a heavy inter-processor communication has been introduced. We will study this method in a forthcoming paper.

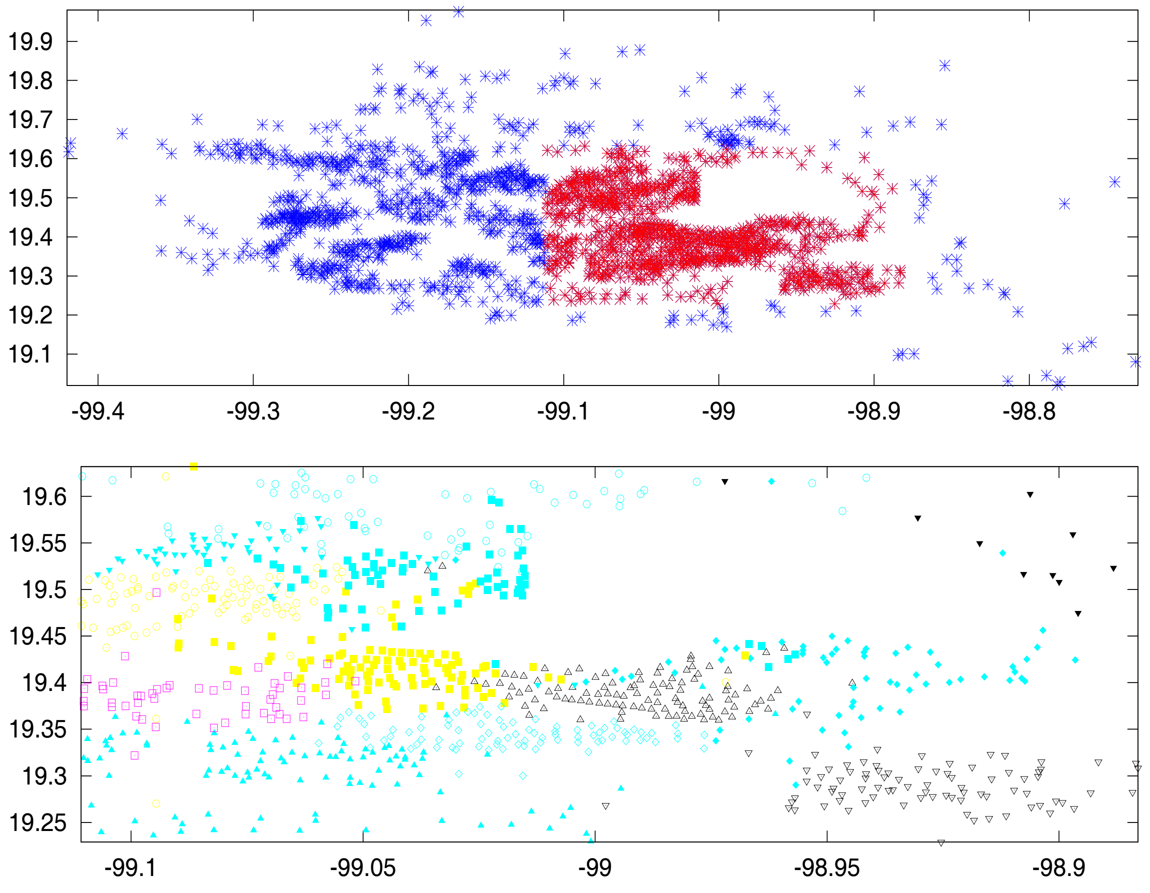


Figure 5. Clustering results for a rectangular area in the valley of Mexico. All demographic variables were used. The plot at the top shows in red the chosen area whereas the bottom plot presents the obtained clusters.

7. Conclusions

In this paper, we have reported the results of an ongoing project aimed to build an open source system for dealing with the zone design problem. In particular, we are interested in demographic data.

The novelty of our approach is based on the use of a graph partitioning technique popular in the area of complex networks research: community detection by using the betweenness centrality metric. We have shown that even a simplified model (we map the problem to an unweighted graph in contrast to a weighted one) is able to capture enough information about the clustering structure of the data.

We are building the system on top of an interactive high-performance computing platform that acts as an interface between parallel hardware of different computational power and the problem software solution. This strategy should make the system useful for researchers and practitioners with low budget and, at the same time, should also be able to control a large cluster. What is important is that in both scenarios the user sees the same enviroment.

Our results confirm two things: 1) the proposed method is a promising alternative to previous efforts in this area; and, 2) the viability of the construction of an open source system with these purposes.

We believe that our approach can be used with similar results in other zone design problems.

References

[1] Bernábe B., Desarrollo de un modelo para la determinación de zonificación óptima. PhD thesis, División de Estudios de Posgrado de la Facultad de Ingeniería DEPFI, UNAM, Investigación de Operaciones, 2006. Proyecto de tesis doctoral.

[2] Duque, J. C., Church, R. L., and Middleton, R. S., Exact models for the regionalization problem, In Western Regional Science Association Annual Meetings, Santa Fe 2006.

[3] Juan Carlos D., Raúl Ramos, and Jordi Suriñach., Supervised Regionalization Methods: A Survey, Vol. 30, International Regional Science Review 2004, pp. 195-220.

[4] Pizza E., Murilo A., Trejos J., Nuevas Técnicas de Particionamiento en Clasificación Automática, Revista de Matemáticas Teoría y Aplicaciones, ISSN 1409-2433, 1999, pp. 51-66.

[5] Bacao F., Lobo V., Painho M., Applying Genetic Algorithms to Zone Design, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 2, Issue 5, ISSN:1432-7643, Springer-Verlag Heidelberg 2005, pp. 341-348.

[6] Bernábe L. B., Ramírez R. J., Espinosa R. J., Evaluación de un algoritmo de recocido simulado con superficies de respuestas, Revista de Matemáticas Teoría y Aplicaciones, Vol. 16, No. 1, ISSN 1409-2433, Enero/Julio 2009, pp. 159-177

[7] Bernábe B. and López R., Proceso de Análisis de Correlaciones para Identificar Patrones de Asociación Sobre Datos Censales, Research on Computing Science, ISSN 1665-989, CIC – IPN 2005.

[8] M. Girvan and M. E. J. Newman., Community structure in social and biological networks, Vol 99, arXiv:cond-mat/0112110 v1, In National Academy of Sciences, December 2002, pp. 7821-7826.

[9] M. E. J. Newman., Analysis of weighted networks. Physical Review E, 70(056131), Nov 2004. arXiv:cond-mat/0407503 v1, 20 Jul 2004.

[10] M. J. E. Newman., Fast algorithm for detecting community structure in networks, Physical Review E, 69(066133), 2004. arXiv:cond-mat/0309508 v1, 22 September 2003

[11] A. Ochoa and L. Arco., Differential Betweenness in Complex Networks Clustering, In Progress in Pattern Recognition, Image Analysis and Applications, 13 Iberoamerican Congress on Pattern Recognition, Havana Cuba, Springer Verglag LNCS 5197, September 2008, pp. 227-234.

[12] Qiaofeng Yang and Stefano Lonardi., A parallel edge-betweenness clustering tool for protein-protein interaction networks, Int. J. Data Mining and Bioinformatics, 1(3), 2007.

[13] O. Ochoa., ViPoC una nueva herramienta de software libre para computación interactiva de alto rendimiento, PhD thesis, ICIMAF. Havana. Cuba, adviser: Alberto Ochoa, 2009.

[14] Omar Ochoa Rodríguez and Alberto Ochoa Rodríguez., ViPoC: un clúster virtual interactivo y portátil, Revista Ingeniería Electrónica, Automática y Comunicaciones, Vol. 27, No. 3, 2007.

[15] Gabor Csardi and Tamas Nepusz., The igraph software package for complex network research, InterJournal Complex Systems. pp. 1695. 2006.

[16] Instituto Nacional de Estadística, Geografía e Informática (INEGI). http://www.inegi.gob.mx.

Authors Biography

Alberto Ochoa

He received the B.Sc. degree in Havana, 1985 and a Ph.D. degree in computer science in Moscow, 1992. Currently, he is a full-time senior researcher at the Institute of Cybernetics, Mathematics and Physics of Cuba.
He is a leading researcher in the field of evolutionary computation and one of the creators of the important class of Estimation of Distribution Algorithms. In this field, in pattern recognition and in machine learning, he has authored or coauthored many works on scientific journals, book chapters and conference proceedings.

He has given invited lectures in many countries of Europe and Latin America and regularly acts as a reviewer of important conferences and journals.

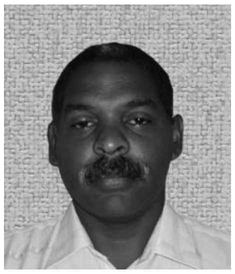
His current research interest is wide: complex networks, information theory, probability and stochastic modeling, graphical models, evolutionary computation, text/data mining, bioinformatics, image analysis and parallel programming.

Beatríz Bernábe

She was born in Puebla, Mexico, in 1966. He received the B.S. degree in computer science from Benemérita Universidad Autónoma de Puebla (BUAP), Mexico, in 1995, the M.I. degree in quality engineering from Universidad Iberoamericana (UIA), Mexico, in 2003. In June 2008, she received the Ph.D. candidate degree from the Universidad Nacional Autónoma de México (UNAM). Since 1995, she has been a professor at the School of Computer Science of BUAP, where she works in databases and statistics.

She is currently working on a PhD in the area of operational research at the Faculty of Engineering, UNAM.

Her academic interests are in the following areas: combinatorial optimization, territorial design and multiobjective techniques.



Omar Ochoa

He received the B.Sc. degree in electrical engineering from the Instituto Superior Politécnico (ISPJAE) in Havana, 1990. He is currently working on a PhD in the area of open source parallel systems at the Institute of Cybernetics, Mathematics and Physics of Cuba.

His academic interests are in the following areas: parallel computation, computer networks, free software and web technologies.