Universidade Federal de Minas
Gerais Instituto de Ciências Exatas
Departamento de Estatística

A Simulated Annealing Strategy for Cluster Detection

Luiz Duczmal
Renato Martins Assunção

Relatório Técnico
RTP-09/2001

Relatório Técnico
Série Pesquisa

ABSTRACT

We discuss and implement a new strategy for spatial cluster detection. A test statistic based on the likelihood ratio is used, as formulated by Kulldorff and Nagarwalla. Differently from these authors, our test is not restricted to the detection of clusters with fixed shape, such as rectangular or circular shape, but it looks for connected clusters with arbitrary geometry. This could be advantageous in real situations, where we frequently find spatial clusters along rivers or transport ways, for example. A new technique of adaptive simulated annealing is developed, focused on the problem of finding the local maxima of a certain likelihood function over the space of the connected subgraphs of the graph associated to the map of populations and geo-referenced cases. This algorithm has applications to the study of disease clusters and hot-spots of criminality. We present a study case for homicides in the city of Belo Horizonte, Brazil.

KEYWORDS: Spatial cluster detection, simulated annealing, likelihood ratio test, disease clusters, hot-spots detection.

1. INTRODUCTION

Epidemiologists and crime analysts have interest in the detection and monitoring of disease clusters or hot-spots of criminality. A rather complete discussion about spatial cluster detection and inference may be found in [1], [3], [4], [6] and [7]. We assume that we have at our disposal a map of regions, each one with a defined risk population and a certain number of observed cases. The cases corresponds to the individuals in each population that have a special designation, such as an infected individual or a crime victim. For simplicity, we may represent the regions by polygons, and the common frontier between two regions are either a single point or a (non-trivial) segment (see fig. 1.a). Two regions are said to be *neighbors* when they are connected by a segment. This map of interconnected regions can be further simplified and mathematically represented by a graph, where each region is associated to a node, and when two regions are neighbors, there is an edge in the graph linking the corresponding two nodes (fig. 1.b). Each node may have a number of attributes, such as the population and the number of cases of the corresponding region. A connected subset of regions of the map is called a *zone* (see, e.g. figure 2). Corresponding to each zone there is a connected *subgraph* of the map graph. From now on we will identify each subgraph with the zone that it represents.

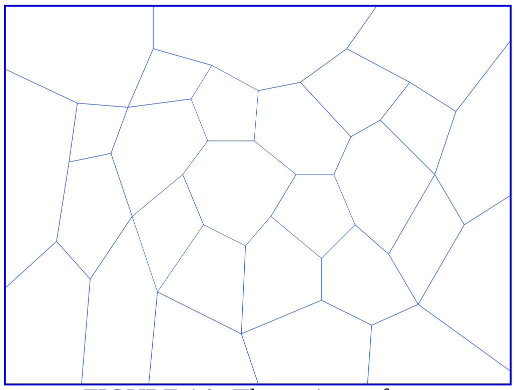


FIGURE 1A: The regions of a map

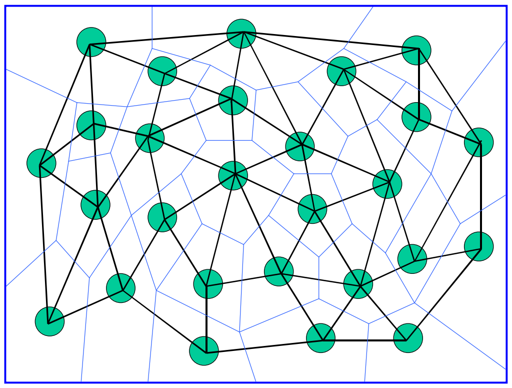


Figure 1B: The associated graph

We consider here the problem of finding, among all zones, which one is the most likely (if any) to have a high incidence of cases relative to the population. We will call such zone a *cluster.* The precise formulation of this problem will be addressed in the next sections.

The algorithm considered in [6] sweeps over the zones circumscribed by circles of varying radii centered at each of the regions of the map. At each one of these zones, a test statistic function is computed, and the algorithm selects the zone that maximizes this function, as the most likely to be a cluster, if it exists. In this paper we will present a graph-based algorithm that overcomes some of the limitations of that algorithm.

The choice of the regions in real maps deserves some attention. We would like to choose regions that are small enough to circumscribe a relatively homogeneous area, in such a manner that we could consider the population and the cases inside each region as roughly similar. If this condition cannot be fulfilled, it may not be possible to consider the attributes of the graph nodes as adequate descriptions of the regions. In this case, it would be necessary to further refine the regions in the original map, in order to create a new set of smaller and more uniform regions. For example, if one single region is big enough to contain aggregates of individuals with very dissimilar incomes, we cannot expect to find homogeneous poverty-related disease rates within this region.

We now proceed to define some terms that will be used in our discussion.

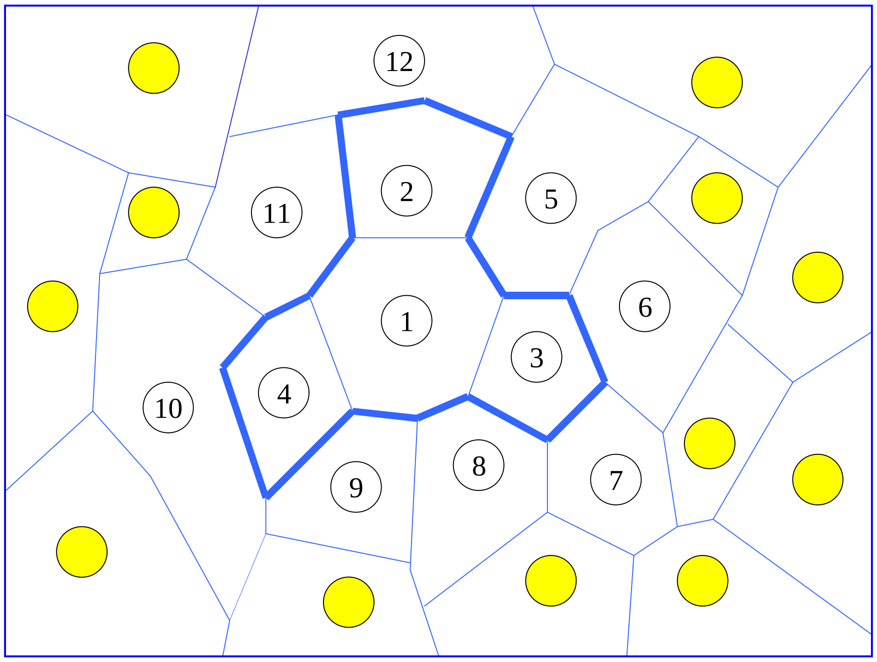


FIGURE 2: A zone within a map

2. THE SETTING OF THE PROBLEM

A *non-oriented graph* (or *graph,* for short) G is an ordered pair (V,E), where V is a finite set of *vertices* $\{v_1,...,v_n\}$ and E is a set of *edges* $\{e_1,...,e_m\}$, such that $e_i = \{v_{i_1}, v_{i_2}\}$, with $v_{i_1} \neq v_{i_2}$ and $v_{i_1}, v_{i_2} \in V$, $i = 1,...,m$. If $\{v_j, v_k\}$ is an edge, then v_j and v_k are *adjacent* vertices. The graph $S = (V_1, E_1)$ is a *subgraph of* $G = (V,E)$ if $V_1 \subset V$, and $E_1 \subset E$. The subgraph $S = (V_1, E_1)$ of $G = (V,E)$ is an *inherited subgraph of* G if $v_j, v_k \in V_1, \{v_j, v_k\} \in E \Rightarrow \{v_j, v_k\} \in E_1$, i.e., E_1 is a maximal set over all the subgraphs (V_1, W) of G.

Two vertices v_j, v_k of a graph $G = (V,E)$ *are connected by a path* if there is a sequence of vertices $v_{r_1}, v_{r_2} ..., v_{r_p}$ such that $v_j = v_{r_1}, v_k = v_{r_p}$, and $\{v_{r_i}, v_{r_{i+1}}\} \in E, i = 1,..., p-1$.
A graph $G = (V,E)$ is *connected* if any pair of distinct vertices $v_j, v_k \in V$ are connected by a path.
The inherited subgraphs $S_1 = (V_1, E_1)$ and $S_2 = (V_2, E_2)$ of $G = (V,E)$ are *neighbors* if the set $(V_1 \cup V_2) - (V_1 \cap V_2)$ consists of exactly one element.
For each connected inherited subgraph S of G, the *connected neighborhood* (or simply the *neighborhood*) $N(S)$ of S is the set of all connected inherited neighboring subgraphs of S.
Consider, for example, the map in figure 2. The zone with the vertices set {1,2,3,4} has 11 connected inherited neighbors, namely the ones with the vertices sets {1,2,3,4,5}, {1,2,3,4,6}, {1,2,3,4,7}, {1,2,3,4,8}, {1,2,3,4,9}, {1,2,3,4,10}, {1,2,3,4,11}, {1,2,3,4,12}, {1,2,3}, {1,2,4} and {1,3,4}. Observe that the neighbor with vertices set {2,3,4} is not connected.

We follow [6] to establish the statistical notation. Let *z* be a cluster candidate among the set **Z** of all connected inherited subgraphs associated to the regions of the map. That is, *z* is a connected subset of regions in the map. Define *p* as the probability that an individual is a case in *z,* and *q* as the probability that an individual is a case in the complement of *z.* We would like to test if *z* is a cluster. The alternative hypothesis is $H_1 : z \in Z, p > q$, and the null hypothesis is $H_0 : p = q$ (i.e., there is no cluster in **Z**). Knowing the population and cases in each region allows us to define n_z as the population of the region *z,* c_z as the number of cases of the region *z,* N as the total population in the map and C as the total number of cases. For this fixed cluster candidate *z* adopting a binomial model produces the following likelihood function:

The likelihood ratio is

$$T = \frac{\sup\limits_{z \in Z, p>q} L(z,p,q)}{\sup\limits_{p=q} L(z,p,q)} \quad (p, q \in [0,1]).$$

The denominator in the expression above reduces to

$$\sup_{p=q} L(z,p,q) = \sup_{p \in [0,1]} p^C (1-p)^{N-C} = \frac{C^C (N-C)^{N-C}}{N^N} = L_0$$

and is a constant that depends only on C.

For a fixed candidate z, we maximize over all possible $0 \le q \le p \le 1$. Let

$$L(z) = \sup_{p>q} L(z,p,q)$$

$$= \begin{cases} \left(\frac{c_z}{n_z}\right)^{c_z} \left(\frac{n_z - c_z}{n_z}\right)^{n_z - c_z} \left(\frac{C - c_z}{N - n_z}\right)^{C - c_z} \left(\frac{N - n_z - (C - c_z)}{N - n_z}\right)^{N - N_z - (C - c_z)}, & if \ \frac{c_z}{n_z} > \frac{C - c_z}{N - n_z} \\ L_0, & otherwise. \end{cases}$$

The objective here is to find the zone z that maximizes the function L. For a map with n regions, we would need to analyze all the connected inherited subgraphs among the 2^n possible subsets of n vertices, and this is a formidable task for a map with hundreds of regions. So we need to try to analyze only the most promising subgraphs of the collection of all connected inherited subgraphs, and discard the less interesting ones.

3. THE ADAPTIVE SIMULATED ANNEALING TECHNIQUE

We will show here an algorithm that makes a sweeping over a subset of the collection of all the connected inherited subgraphs, moving at each step from a subgraph to one of its neighbors, until we find the cluster (or give up the search). We need to establish a set of rules telling us how to choose the best neighbor at each step, in order to try to minimize the number of examined subgraphs. We will also need to make a test to show the significance of the possibly found cluster.

Perhaps the first idea that comes to mind would be to choose always the neighbor subgraph with the highest value of the function L at each step, thus conducting us finally to the cluster some steps ahead. As tempting as it seems, this strategy does not work in general, because it frequently leads us to subgraphs that are only local maxima of the function L, but do not maximize L over all the possible subgraphs. A great improvement could be made to this idea, however, if we allow the algorithm, at judicious times, to randomly choose a neighbor subgraph, instead of picking up the highest L-valued one. Thus, most of the times the algorithm decides for the highest L-valued neighbor, but on some occasions it adopts a less deterministic decision. The effect of this rule changing would be to give the algorithm more freedom to survey adjacent neighborhoods that are potentially more interesting, and that almost never would be visited otherwise. We associate with the degree of determinism of the neighbor choice the physical notion of temperature; the higher the temperature, the increased randomness involved in the selection of the next neighbor.

Instead of using a continuum of temperatures, our algorithm uses only three levels of temperature, high, medium and low, corresponding to three different criteria of choosing the next neighbor of the subgraph:

- High temperature: Uniform random choice of neighbors.
- Medium Temperature: Random choice with chances proportional to the logarithm of the likelihood ratio of the neighbors.
- Low Temperature: Always choose the neighbor with the highest likelihood ratio.

The effects of this strategy in the sweeping of the collection of subgraphs are described below:

- High temperature: Higher mobility, do not have a strong preferential direction.
- Medium Temperature: Has a higher probability of choosing a direction with high likelihood ratio, but without discarding another directions.
- Low Temperature: Deterministic, always choosing the neighbor with the highest likelihood ratio.

In order to unify these three strategies, we create a function ***F(G, temp)*** that receives the current subgraph ***G*** and returns a neighbor of ***G*** chosen accordingly to the choice strategy at the ***high, medium*** or ***low*** temperature value set to the variable ***temp***.

In addition to these three levels of randomness in the process of neighbor selection, we will go further and create one more strategy. Let us define the function ***H(G)*** as follows:
If a vertex v_0 was added to ***G*** in the previous step, the function ***H(G)*** returns a neighbor of ***G*** that includes v_0 and also a randomly chosen extra vertex v_1, adjacent to v_0. If otherwise a vertex v_0 was excluded on the previous step, then returns the neighbor given by ***F(G, low)***. For example, referring again to figure 2, if the previous subgraph has the set of vertices {1,2,4} and the current subgraph has the set of vertices {1,2,3,4}, then the result of ***H(G)*** is chosen randomly among the subgraphs with the set of vertices {1,2,3,4,5}, or {1,2,3,4,6}, or {1,2,3,4,7}, or {1,2,3,4,8}. The objective of successive applications of ***H*** is to try to force the appearance of potentially interesting subgraphs that are normally beyond the range of the sweeping given by the function ***F*** alone. As we shall see, the function ***H*** will be used exclusively when there are indications that the current subgraph is a promising one.

We identified four factors that are relevant to the convenient selection of the strategy at each step, when the algorithm computes the L-function for each one of the neighbors of the current subgraph, and prepares to choose the next neighboring subgraph:

- There was found $(hL = 1)$ or not $(hL = 0)$ a neighbor with higher L-value at the current step;
- The number cs of consecutive steps such that weren't found new subgraphs with L-value > 1.
- The number vb of times that the current subgraph has been visited before in the survey;
- The number cv of common vertices between the current subgraph and the highest yet L-valued one in the survey.

The four parameters *hL, cs, vb* and *cv* are used to modify dynamically the process of selection of the successor of the current subgraph or possibly give up the survey at each step as follows:

```
select randomly an initial connected inherited subgraph G;
do{
    find the set N(G) of all the connected inherited subgraphs neighbors of G;
    compute L for all new subgraphs in N(G);
    compute hL, cs, vb, cv, cs_threshold and cs_threshold_2;
    if (cs>cs_threshold_2) G:= F(G, high) ;
    else{
        if ((hL=0) and (vb>vb_threshold_2)) G:= F(G, medium);
        else if ((hL=0) or (vb>vb_threshold_2)) G:=F(G, low);
              else G:=H(G);
    }
 }while ((cs<=cs_threshold) and (vb<=vb_threshold));
```

In the basic survey algorithm above, the thresholds variables are defined as follows:
cs_threshold=cv
cs_threshold_2=cv/2
vb_threshold is fixed and was empirically determined, and is around 10 for most situations.
vb_threshold_2= vb_threshold /2

The idea of the *basic survey routine* above is to adopt at each step one of four types of choices strategies for the successor of the current subgraph, with different levels of randomness. In order of the most random to the most deterministic, there are the ***F(G, high), F(G, medium)*** and ***F(G, low)*** strategies and at last the ***H(G)*** function strategy. The choice of which strategy is to be used is based on the values of the parameters ***hL, vb*** and ***cs***, that are indicating if the current subgraph ***G*** is becoming more or less promising as the survey goes on.

Thus ***F(G, high)*** is adopted if the current subgraph has a relatively low L-value, was visited many times, and for several steps of the survey the L-values for the subgraphs have not increased.
F(G, medium) is used if the current subgraph has a relatively low L-value, has been visited many times, but there have been an increase of the L-values for some recently surveyed subgraph.

F(G, low) is used if there have been an increase of the L-values for some recently surveyed subgraph, but at least one of the following conditions are true: the current subgraph has a relatively low L-value, or it has been visited many times.

Finally, ***H(G)*** is applied when the current subgraph has a relatively high L-value, has not been visited many times, and there have been an increase of the L-values for some recently surveyed subgraph.

The basic survey routine is finally abandoned when one of the parameters ***vb*** and ***cs*** exceeds the thresholds defined within the algorithm.

The basic survey routine is repeated several times with different initial subgraphs, until the maximum L-value found does not increase for a conveniently long sequence of visited subgraphs, or the storage space for the list of visited subgraphs is exhausted.

After the candidate cluster is found, we make a test to evaluate its statistical significance. Under the null hypothesis, we simply distribute randomly the cases over each region z_i of the cluster, i.e., we use a Poisson random variable with mean $\frac{Cn_{z_i}}{N}$, and compute $L(z)$. We make this thousands of times and count how many times $L(z)$ is greater than the candidate cluster L-value, and this is used to compute an experimental p-value.

4. EXPERIMENTAL RESULTS

We will present some experimental results on the performance of the algorithm described in section 3. For the purpose of simplicity and uniformity in our exposition, we establish a standard map corresponding to the graph realization depicted in figure 3a. It consists of 625 vertices disposed in a 25×25 square. Each internal vertex has 6 adjacent vertices, as shown in the figure. We also define another square $n \times n$ maps with different sizes and with vertices and edges defined in a similar fashion, and will call them *standard* $n \times n$ *maps*. A *standard* $m \times m$ *cluster* within a standard map would be a $m \times m$ square centered subgraph. Figure 3a also shows a standard 5×5 cluster within the standard 25×25 map. To each vertex in the standard map we associate a population of 100 individuals. The number of cases is $10 + w$ for vertices within the standard cluster, and $1 + w$ outside it, where w is an uniform random variable, an integer random number $w \in [0, u]$, and u is a non-negative integer. We say that w is an additive noise of level u. Of course, due to the random choice of the cases in each vertice, the final cluster found by the algorithm may be somewhat different from the 5×5 square, as shown in the typical example of figure 3b. Generally speaking, more additive noise generates more obliterated clusters. Another types of clusters will also be discussed (See e.g. figures 3c-f). The results shown here may be extended with few modifications to another types of maps. The standard map is sufficiently complex to exhibit several interesting features, that we will describe now.

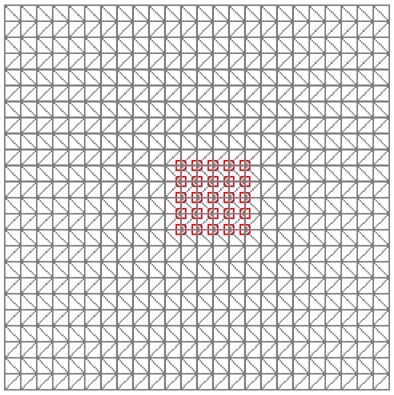


FIGURE 3A

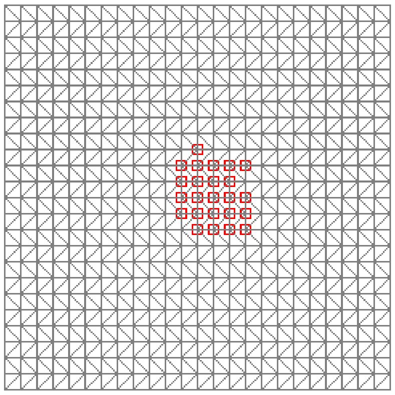


FIGURE 3B

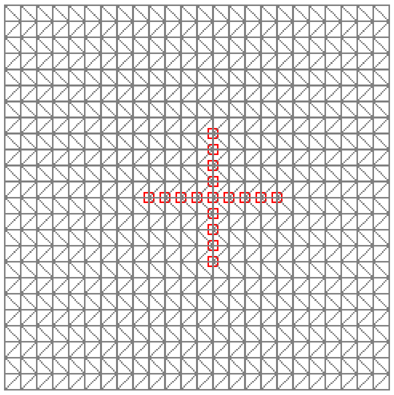


FIGURE 3C

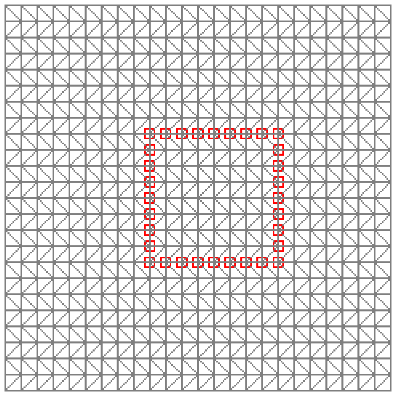


FIGURE 3D

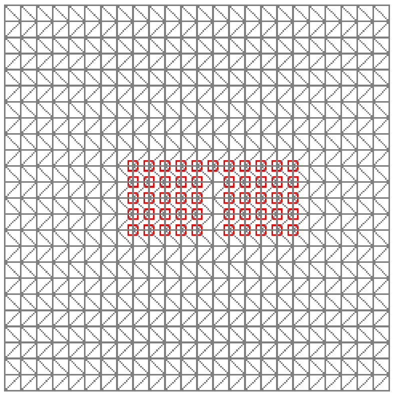


FIGURE 3E

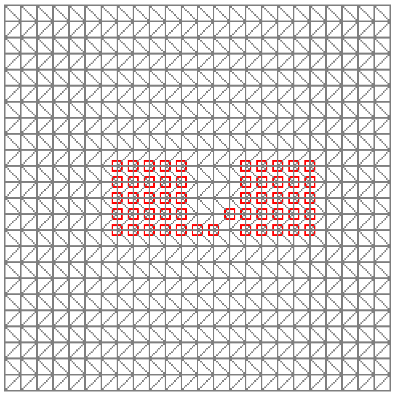


FIGURE 3F

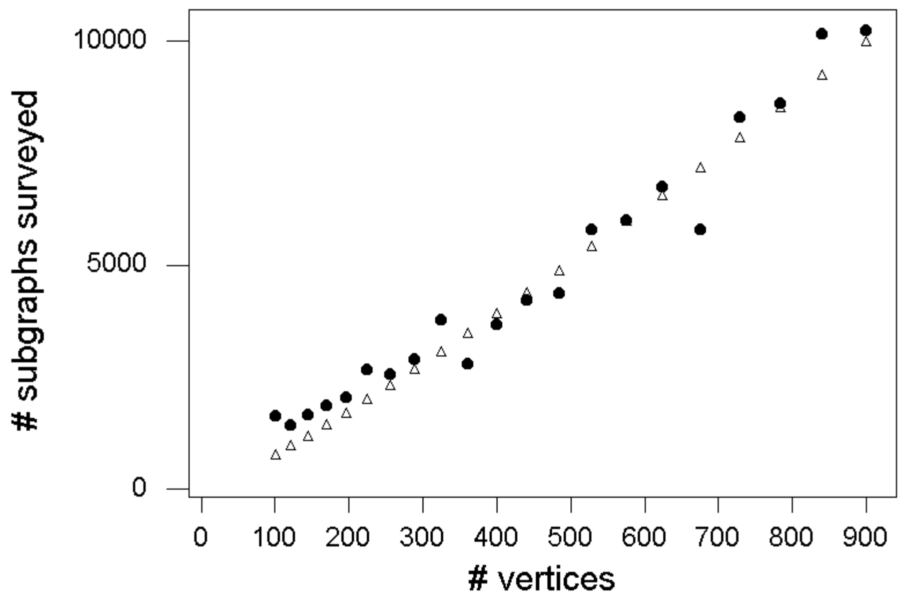


FIGURE 4: Performance of the algorithm when the size of the map increases, for the 5X5 standard cluster.

The solid circles in figure 4 denotes the number of surveyed subgraphs for each standard $n \times n$ map with $k = n^2$ vertices with a standard 5×5 cluster. The additive noise is of level 5. Each experimental point (solid circle) here is obtained as the median of the averaged number of surveyed subgraphs in 10 runs of the algorithm for each of five different random standard maps. The triangles denotes the values $c\,k\ln(k)$, with $c = 1.63$, for comparison. We will return to this later on section 5.

When the additive noise is zero, the mean number of visited surveyed subgraphs (circles in the figure 5) increases appreciably. But the mean number of analyzed surveyed subgraphs for the standard maps (crosses), increases only slightly. This occurs because the basic survey routine gives up more easily when the noise level is zero, thus reducing the number of analyzed subgraphs.

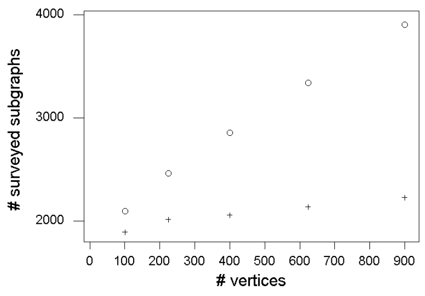


FIGURE 5: The influence of noise

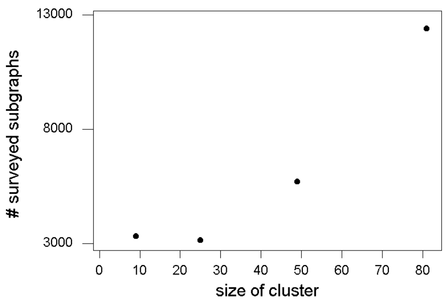


FIGURE 6: The influence of cluster size

Figure 6 shows the mean number of surveyed subgraphs for the standard map with 625 vertices, for some standard clusters of different sizes. As we can see in this figure, the mean number of surveyed subgraphs seems to increase with the size of the cluster, for clusters with 25 vertices or more.

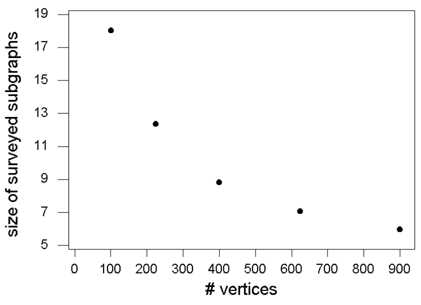


FIGURE 7A: The average size of surveyed subgraphs

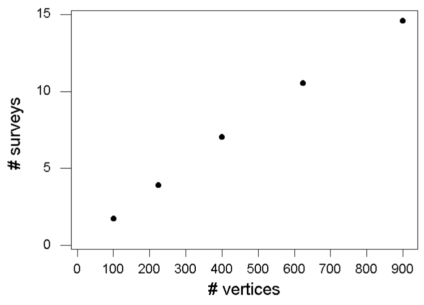


FIGURE 7B: The average number of basic surveys

We can see in figure 7a that the mean size of the surveyed subgraphs decreases with the number of vertices of the standard map and standard 5×5 clusters with noise level 5. This happens because the algorithm gives up more easily when it searches through larger “empty” regions outside the cluster within the map. The basic survey routine is called more times for bigger maps, as we can see in figure 7b.

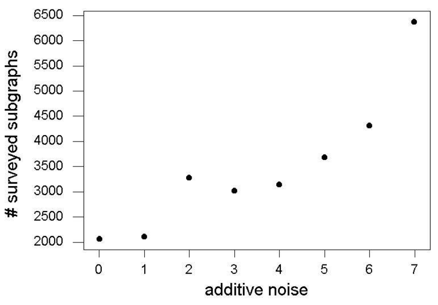


FIGURE 8: Noise affects the performance

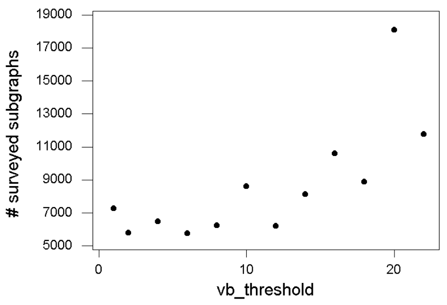


FIGURE 9: Finding the optimal value of vb_threshold

The presence of additive noise has influence on the mean number of surveyed subgraphs, but in a rather complex manner, as we can see in figure 8. We observe that there is a sudden increase in the mean number of surveyed subgraphs, when the noise level increases above 1. At this point, large zones with more or less the same value appear in the map, and the algorithm takes some time searching them until they are discarded. This effect is less pronounced in maps with noise level 3 and 4. Beyond this point, as the noise level increases, there is a steady increase in the mean number of surveyed subgraphs.

In order to evaluate experimentally the best value for the parameter *vb_threshold*, we conducted a test with the 625 vertices standard map with the standard 5×5 cluster with noise level 5. As we can see in figure 9, the choice for the parameter *vb_threshold* is not very critical. We usually adopted the value 8 in our tests, because it is also apparently better for other more complicated maps.

The cluster of figure 3c, with noise level 5, was found after analyzing 2,275 subgraphs, and the one of figure 3d, in the same conditions, was found after 6,173 subgraphs. The performance of the algorithm for these two cases are comparable with the performance for the standard clusters of equivalent size. Note that these two cases could not have been found by the original circle centered zones algorithm in [6].

Consider now the case where we have two separated 5×5 blocks, the "double cluster" examples. Figures 3e and 3f depict the clusters found when the algorithm tries to analyze them. In figure 3e, the two blocks are isolated by one region, and in figure 3f they are three regions apart. In both cases the algorithm finds first one of the isolated blocks (after 741 and 842 searches, respectively), and then, after a longer search, it finds the clusters formed by joining them (after 5,736 and 11,226 searches, resp.).

As a last example, we analyzed in figure 10 a map of the 2 million inhabitants city of Belo Horizonte, Brazil, with 240 regions, with homicide cases during the year of 1995. North is up in this map. The most significant cluster is marked in the map. The light-gray regions have exactly zero cases, but they are included in the cluster by the algorithm because they link "smaller clusters" with high incidence rates. We can see that the light-gray region in the bottom of the map is specially large in comparison with the others. This is a nuisance in the original planning of the map, because this large region limits with zones that are far away from the high incidence regions near the center of the map. The best solution here, as commented in the introduction, would be to split it in several smaller regions. Anyway, we can see very clearly a north-south almost straight cluster following a large express way in Belo Horizonte, that is known to be a degraded urban area with high incidence of homicides.

This algorithm was implemented in C language, on a Linux environment, on a K6-II 500 MHz processor microcomputer.

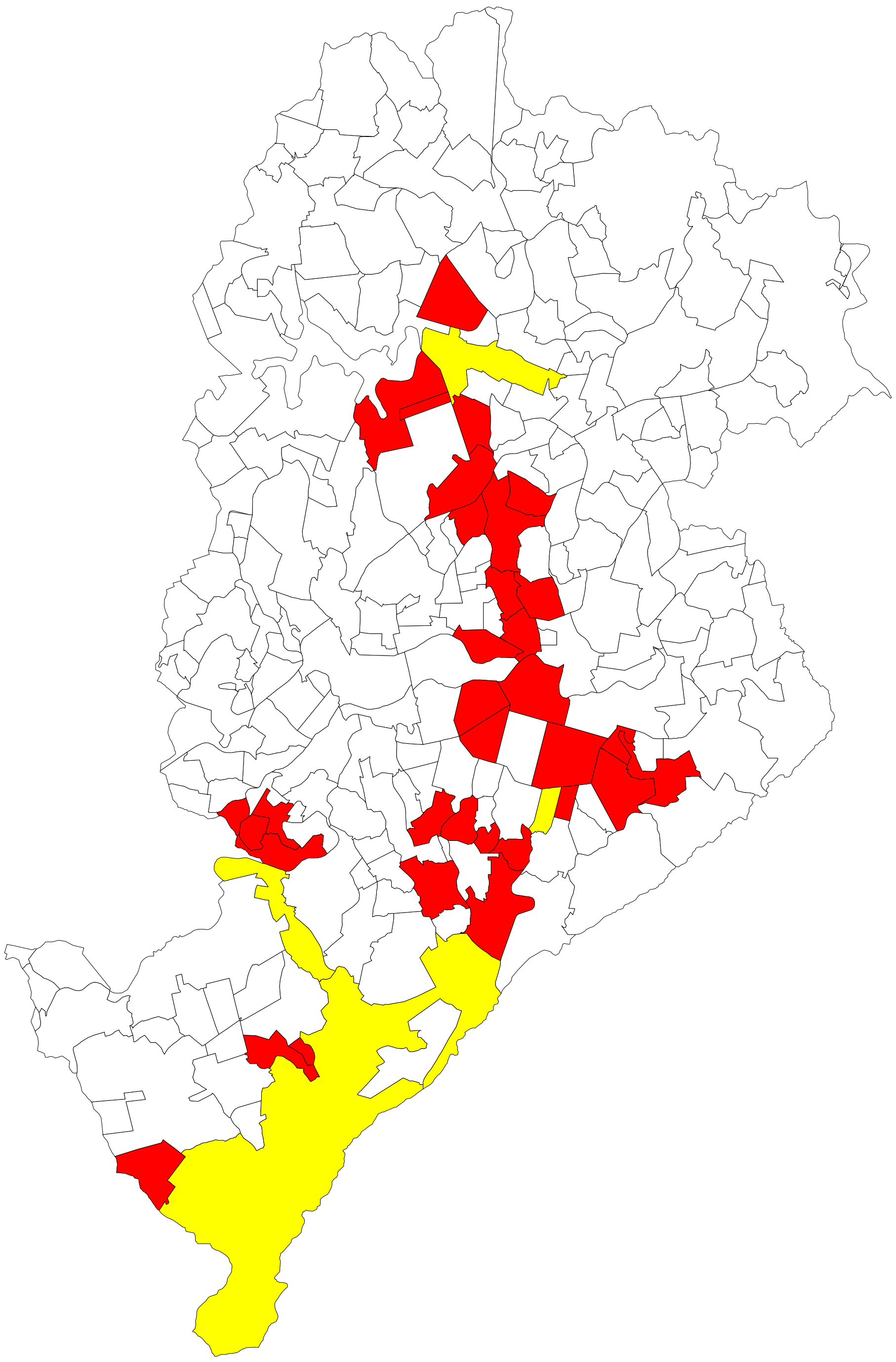


FIGURE 10: The homicide cluster in the city of Belo Horizonte

5. SOME COMMENTS ON THE CONVERGENCE OF THE ALGORITHM

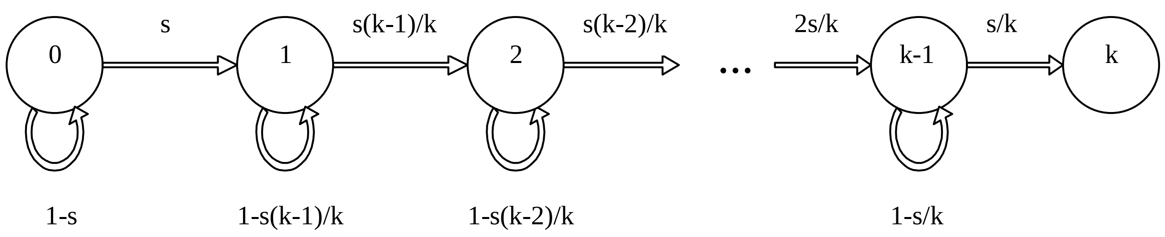
It is a well-know fact that combinatorial algorithms that use simulated annealing converge to the optimal solution in exponential time in the worst case, but usually find quasi-optimal solutions in much less time (see [9]).
The process of section 3 of finding the cluster (if it exists) can be described as a stochastic process as follows. Suppose that to each vertex $v_i, i = 1,...,k$ of the map graph is assigned a binary variable b_i, such that $b_i = 1$ if v_i is in the set of vertices of the current subgraph, and $b_i = 0$ otherwise. Define also the binary constants c_i, such that $c_i = 1$ if v_i is in the set of vertices of the cluster, and $c_i = 0$ otherwise. Thus, starting at an initial variable string $(b_1,...,b_k)_o$ we attempt to reach at the string $(c_1,...,c_k)$, modifying randomly at most one value b_i at each step, and producing a sequence of strings $(b_1,...,b_k)_0$, $(b_1,...,b_k)_1$, ... , $(b_1,...,b_k)_r = (c_1,...,c_k)$ after r steps. Suppose, for simplicity, that the variable b_i once set to the value c_i, maintain this value during the rest of the sequence of strings. Suppose also that when, at each step, an index i is randomly chosen there is a fixed probability s that b_i assumes the value c_i. So the expected mean value of r is approximately $s^{-1}k\ln(k)$, for large k. See the appendix for a proof. Of course, the assumptions above are somewhat artificial. The parameter s may be very difficult to establish. It may be not constant over the map, and also may vary greatly from one problem to another. But this crude model can explain, in certain simple situations, the behavior of the algorithm, as shown in the graph of figure 4. In this particular case the cluster is very well defined in the map, and the algorithm does not need to worry with small scattered “clusters”. This is certainly not the case as with the cluster of the city of Belo Horizonte, or in the “double cluster” example.

The experiments in section 4 suggests that the worst scenario for the algorithm is the presence of several disconnected “small clusters” scattered along the map. The algorithm finds easily each one of these “small clusters”, but only after a lengthy survey it finds the increasing bigger “clusters” made by joining these “small clusters” together. So, the “clumpyness” of the map, loosely defined here as the presence of these scattered disconnected “small clusters”, is a factor of importance in the performance of the algorithm. It could also leads us to a discussion of the original concept of cluster. Perhaps it would be useful to consider not only the cluster, but also the possible combinations of “small clusters” found during the course of the survey.

If we allow that the parameter s discussed above has a different value for each region of the map, we can understand better some aspects of the behavior of the algorithm in the "double cluster" examples of figures 3e and 3f. In figure 3e, for example, we adopt very small values for the parameter s for the five vertices between the two 5×5 blocks. It means that it is much more difficult for the algorithm to recognize that one of these five vertices are part of the cluster. A similar reasoning applies to the fifteen vertices between the two 5×5 blocks in figure 3f.

6. APPENDIX

Theorem: The stochastic process described by the graph below



is such that the average number of transitions from state 0 to state k is asymptotically given by $s^{-1}k\ln(k)$.
Proof: Let A be the corresponding stochastic $k+1\times k+1$ matrix with entries

$$a_{ij}=\begin{cases} s(i-1)/k, & i=j \\ 1-s(i-1)/k, & i+1=j \\ 0, & otherwise. \end{cases}$$

Consider now the $k\times k$ matrix B obtained from A suppressing the last line and the last column. Each entry w_{ij} of the matrix $W=(I-B)^{-1}$ is the average number of times that the state j appears in a random path from the state i to the state j (see, e.g. [8]). A computation shows that

$$w_{ij}=\begin{cases} s^{-1}k/(k-j+1), & i\le j \\ 0, & i>j. \end{cases}$$

It is now easily seen that the average number of transitions to reach the state k from the state 0 is given by
$$s^{-1}k\sum_{j=1}^{k}\frac{1}{j},$$
the sum of the entries of the first line of W.
Now, using the fact that

$$\sum_{j=1}^{k}\frac{1}{j}=\ln(k)+\gamma+\frac{1}{2k}+O\left(\frac{1}{k^2}\right),$$

where $\gamma=0.577...$ is the Euler constant, the result follows.

7. REFERENCES

.
[1]Banfield, J. D. e Raftery, A. E. (1993) Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49, 803-822.
[2]Friedman, H.P. e Rubin, J. (1967) On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62, 1159-1178.
[3]Fukada, Y. (1980) Spatial clustering for region analysis. *Pattern Recognition*, 12, 395-403.
[4]Huel, G., Petiot, J. F., Lazar, P. (1986) Algorithm for the grouping of contiguous geographical zones. *Statistics in Medicine,* 5, 171-181.
[5]Knorr-Held, L. e Raer, G. (1999) Bayesian Detection of Clusters and Discontinuities. *Preprint*
[6]Kulldorff, M. and Nagarwalla, N. (1995) Spatial Disease Clusters: Detection and Inference. *Statistics in Medicine*, 14, 779-810
[7]Scott, A. J. e Symons, M. J. (1971) Clustering methods based on likelihood ratio criteria. *Biometrics*, 37, 35-43.
[8]Kemp, R. (1984) Fundamentals of the Average Case Analysis of Particular Algorithms, *Wiley-Teubner Series in Computer Science*
[9]Winkler, G. (1995) Image Analysis, Random Fields and Dynamic Monte Carlo Methods. *Springer*