

## **Efficient regionalisation techniques for socio-economic geographical units using minimum spanning trees**

RENATO MARTINS ASSUNÇÃO<sup>1</sup>

MARCOS CORRÊA NEVES<sup>2</sup>

GILBERTO CÂMARA<sup>3\*</sup>

CORINA DA COSTA FREITAS<sup>3</sup>

<sup>1</sup>Federal University of Minas Gerais (UFMG),  
Department of Statistics,  
Av. Antônio Carlos, 6627 - Pampulha  
31270-901, Belo Horizonte, MG, Brazil  
Tel: (031) 3499 5920 , Fax: (031) 3499 5924  
[assuncao@est.ufmg.br](mailto:assuncao@est.ufmg.br)

<sup>2</sup>Brazilian Agricultural Research Corporation (EMBRAPA),  
Nacional Centre for Environmental Monitoring (CNPMA),  
P.O. Box 69, 13820-000 Jaguariúna, SP, Brazil  
Phone: +55-19- 38678700  
[marcos@cnpma.embrapa.br](mailto:marcos@cnpma.embrapa.br)

<sup>3</sup>National Institute of Space Research (INPE),  
Image Processing Division (DPI),  
P.O.Box 515, 12227-001, São José dos Campos (SP), Brazil.  
Phone: +55-12-39456499 Fax: +55-12-39456460  
[gilberto@dpi.inpe.br](mailto:gilberto@dpi.inpe.br); [corina@dpi.inpe.br](mailto:corina@dpi.inpe.br)

---

\* Corresponding author.

***Abstract***

Regionalisation is a classification procedure applied to spatial objects with an areal representation, which groups them into homogeneous contiguous regions. This paper presents an efficient method for regionalisation. The first step creates a connectivity graph that captures the neighbourhood relationship between the spatial objects. The cost of each edge in the graph is inversely proportional to the similarity between the regions it joins. We summarise the neighbourhood structure by a minimum spanning tree (MST), which is a connected tree with no circuits. We partition the MST by successive removal of edges that link dissimilar regions. The result is the division of the spatial objects into connected regions that have maximum internal homogeneity. Since the MST partitioning problem is NP-hard, we propose a heuristic to speed up the tree partitioning significantly. Our results show that our proposed method combines performance and quality and it is a good alternative to other regionalisation methods found in the literature.

**Keywords:** regionalisation, constrained clustering, graph partitioning, optimisation, zone design, census data analysis.

## 1 Introduction

In a significant number of geographical applications, such as socio-economic, health and census data analysis, data is organised as a large set of spatial objects represented by areas. Examples of these spatial objects include census tracts, health districts and municipalities. In many circumstances, it is desirable to group a large number of spatial objects into a smaller number of subsets of objects, which are internally homogeneous and occupy contiguous regions in space. This procedure is called *regionalisation*, which results in new areas (called *regions*) with a more comprehensive geographic extent. The idea of regionalisation applied to socio-economic units (also called *zone design*) was pioneered by Openshaw (1977). Grouping a set of homogeneous areal units to compose a larger region can be useful for sampling procedures (Martin, 1998). Regionalisation is especially valuable for dealing with large data sets of areal units, removing fine scale variations and preserving relevant patterns. If properly carried out, it can produce entities that are more useful for spatial pattern discovery than the original data set (Openshaw and Alvanides, 1999).

There are three types of regionalisation techniques proposed in the literature. The first one is a two-step procedure that applies a nonspatial clustering algorithm, followed by a neighbourhood-preserving classification. The second one uses the geographical coordinates of the location as extra attributes in the clustering procedure (Haining et al., 2000). In these two techniques, the clustering procedure is not constrained by the neighbourhood relationship between the areal units, being introduced either as a second step in the method or as an indirect constraint when using coordinates as attributes. Therefore, the resulting clusters may not properly depict the inherent spatial patterns of the data set (Openshaw and Wymer, 1995). In the third approach, the

neighbourhood relationship between the spatial objects is used explicitly in the optimisation procedure. The best-known approach to regionalisation of socio-economic units based on the latter approach is the AZP algorithm (Openshaw and Rao, 1995). While the original AZP algorithm used ‘Monte Carlo’ optimisation, Alvanides et al. (2002) developed an alternative implementation of AZP using simulated annealing.

In this paper, we follow the third approach, aiming for an efficient regionalisation procedure. We represent the objects by a graph, and our proposed algorithm prunes the graph to get contiguous clusters. The advantage of graph-based techniques is to make spatial adjacency an essential part of the clustering procedure. The challenge for graph-based regionalisation techniques is twofold: to bring down the computational cost of the optimisation procedure and to reduce the sensitiveness on the choice of the initial position for tree partitioning. To address these questions, we present a technique for graph-based regionalisation in two steps. In the first step, we create a *minimum spanning tree* (MST) from the neighbourhood graph. This MST represents a statistical summary of the neighbourhood graph. Then we employ a heuristic to prune the MST and get a set of spatial clusters. In the paper, Section 2 presents a review of the literature. Section 3 describes the proposed method, showing how to build the MST and describing how to partition the tree efficiently. In Section 4 we apply the method on a case study on the city of São Paulo. Section 5 presents a performance analysis in comparison with AZP method.

## **2 Regionalisation approaches: a review of the literature**

There are three main approaches to regionalisation. The first approach is a two-step algorithm. In the first step, a conventional clustering procedure is performed using

nonspatial attributes. In the second step, objects in the same cluster with no spatial contact will be split, forming different regions. This solution enables a quick evaluation of spatial dependence among objects. However, this method does not capture the spatial adjacency condition directly, resulting in limited capacity to capture spatial patterns. Another inconvenience of this approach is the lack of control on the resulting regions. Data sets with low spatial autocorrelation will result in more regions than desirable (Haining et al., 2000).

The second approach used in regionalisation considers both the geographical position and nonspatial features of those objects. The clustering algorithm uses the coordinates of the area's centroid as extra attributes, and measures similarity between objects as a weighted mean of nearness in the feature space and nearness in the geographical space. To work well, the algorithm needs a choice of weights that produces connected clusters. The analyst has to perform the procedure several times, trying different weights until she gets connected regions. This weighted means approach is used by the regionalisation method of the SAGE system (*Spatial Analysis in a GIS Environment*) (Haining et al., 2000). SAGE employs an iterative *k-means* clustering procedure whose objective function has three terms: a) *homogeneity*: similarity in the feature space; b) *compactness*: nearness in the geographical space, based on centroid coordinates; c) *equality*: similarity of values of a selected attribute in the resulting clusters (population, for example). SAGE employs the *equality* measure to produce balanced regions for a given attribute, to allow proper comparisons across resulting regions. Typical control attributes are total population or population at risk for a given disease. Martin (1998) uses a similar approach to produce output areas for the analysis of UK census data.

Openshaw et al.(1998) criticise the weighed means regionalisation algorithm, since the three conditions (homogeneity, compactness, and equality) are not strictly comparable. According to them, a better and simpler strategy would be to select homogeneity as the sole basis for the objective function. Compactness and equality are constraints, and should not be combined with homogeneity measures. Instead, compactness and equality should be handled by defining their expected values (example: minimum population within a region).

The third approach includes algorithms that use adjacency relations as constraints to the clustering procedure. The AZP (*Automatic Zoning Procedure*), proposed by Openshaw (1995) is an example. This method starts with performing a random partition of  $n$  objects in  $k$  regions. Then, through trial and error, it seeks to reallocate objects over regions to minimise an objective function, subject to the adjacency constraint. Improvements to AZP led to the ZDES automatic zoning system (Alvanides et al., 2002). Since the AZP algorithm is computationally expensive, it is useful to consider other techniques that use the adjacency relations of the objects and are computationally efficient. This interest leads to the authors' proposal, described in the next section.

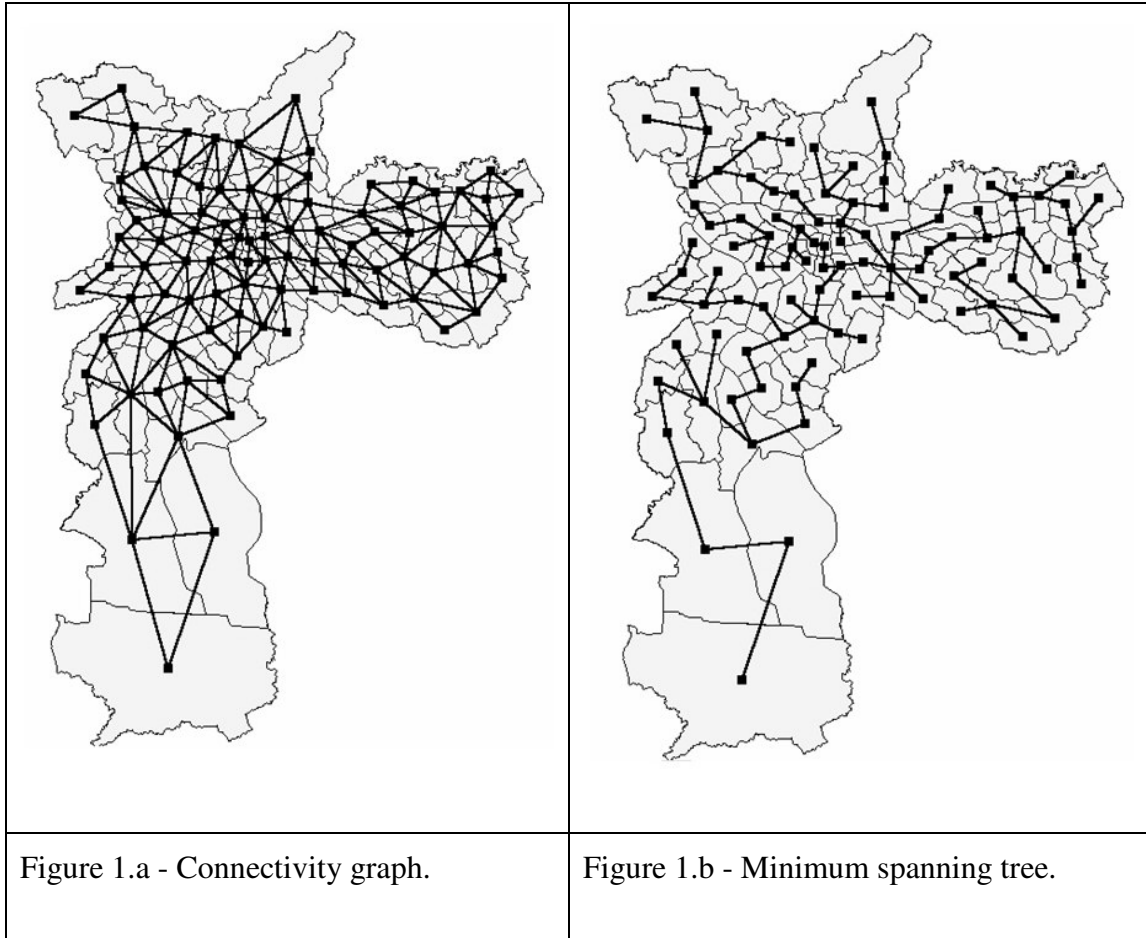
### **3 Regionalisation through graph partitioning**

#### **3.1 General strategy**

In this section, we describe our strategy for transforming the regionalisation problem into a graph partitioning problem. We refer to the algorithm as SKATER (*Spatial 'K'luster Analysis by Tree Edge Removal*). We use a *connectivity graph* to capture the adjacency relations between objects, as shown in Figure 1a. In the graph, each object is associated to a vertex and linked by edges to its neighbours. The cost of each edge is

proportional to the dissimilarity between the objects it joins, where we measure dissimilarity using the values of the attributes of the neighbouring pair. By cutting the graph at suitable places, we get connected clusters. In this way, we transform the regionalisation problem in an optimal graph partitioning problem. Since optimal graph partitioning is an NP-hard problem, we need a heuristic applicable to large spatial data sets to achieve suboptimal solutions in acceptable time. For a general discussion on heuristics for graph partitioning for other types of application, see Even et al (1997) and Karypis and Kumar (1998).

Our proposal is to limit the complexity of the graph by pruning edges with high dissimilarity. The pruning produces a reduced graph that defines a smaller class of possible partitions and whose edges join similar areas. In the reduced graph, further removal of any edge splits the graph into two unconnected subgraphs. To carry out this idea, we take our reduced graph to be a *minimum spanning tree (MST)*, defined in the next section and shown in Figure 1b. After creating the MST, we get clusters by partitioning the tree. Since the optimal solution for partitioning an MST is also an NP-hard problem, we propose a heuristic procedure for tree partitioning that produces good quality clusters at acceptable computational costs. We describe the method in detail below.



### 3.2 Definition of an MST

In this section, we define a minimum spanning tree and discuss the conditions for its uniqueness. Consider a set of areal spatial objects  $O$  with a set of attributes  $\{A_1, \dots, A_n\}$ . All objects have an *attribute vector*  $\mathbf{x} = (a_1, \dots, a_n)$  where  $a_i$  is a possible value of the attribute  $A_i$ . The topology of the set determines a connectivity graph  $G = (V, L)$  with a set of vertices  $V$  and a set of edges  $L$ . There is an edge connecting vertices  $v_i$  and  $v_j$  if areas  $i$  and  $j$  are adjacent (see Figure 1A). We associate a *cost*  $d(i, j)$  to the edge  $(v_i, v_j)$  by measuring the dissimilarity between objects  $i$  and  $j$  using their attribute vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . The dissimilarity measure is context dependent. When the attributes have comparable scales, such as when they are measured in standard deviation units, a usual choice is the square of the Euclidean distance between the attribute vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  :



$$d(i, j) = d(x_i, x_j) = \sum_{l=1}^n (\mathbf{x}_{il} - \mathbf{x}_{jl})^2 \quad (1)$$

A path from node  $v_1$  to node  $v_k$  is a sequence of nodes  $(v_1, v_2, \dots, v_k)$  connected by edges  $(v_1, v_2), \dots, (v_{k-1}, v_k)$ . A graph  $G$  is connected if, for any pair of nodes  $v_i$  and  $v_j$  there is at least one path connecting them. A spatial cluster is a connected subset of nodes. Our aim is to partition the graph  $G$  into  $C$  disjoint spatial clusters  $G_1, \dots, G_C$ , where their union is  $G$  and each one of them is a connected subgraph. This is an NP-hard optimisation problem whose objective function is based on a measure of within-cluster homogeneity.

A *circuit* is a path where the first and the final nodes are the same, and a *tree* is a connected graph with no circuits. A *spanning tree*  $T$  of a graph  $G$  is a tree containing all  $n$  nodes of  $G$ , where any two nodes of  $G$  are connected by a unique path and the number of edges in  $T$  is  $n - 1$ . The removal of any edge from  $T$  results into two disconnected subgraphs that are spatial clusters candidates. A *minimum spanning tree* is a spanning tree with minimum cost, where the cost is measured as the sum of the dissimilarities over all the edges of the tree. The minimum spanning tree is unique if the costs between any node and all its neighbours are distinct (Aho et al., 1983). Should the dissimilarity measure use categorical attributes, many edges would have the same cost and this could lead to more than one minimum spanning tree. In spatial applications involving socio-economic units, we expect the attributes to be random variables with continuous variation, resulting in a unique minimum spanning tree for the usual choices of dissimilarity measures. The algorithm used to build the minimum spanning tree makes this point clear.

### 3.3 MST generation

In this section, we describe the algorithm for building the minimum spanning tree. We build the MST in a recursive way, based on Prim's algorithm (Jungnickel, 1999). Given a connectivity graph  $G = (V, L)$  with a set of vertices ( $V$ ) and a set of edges ( $L$ ), the algorithm starts with one  $T_1$  tree, containing only one vertex. At each iteration, we add a new edge and a new vertex to the tree. In iteration  $n$ , the  $T_n$  tree contains all  $n$  vertices of  $V$  and a subset  $L_m$  of  $L$  with  $n-1$  edges. The sum of costs associated to edges in  $L_m$  is minimal. The steps for MST generation are:

**Step 1:** Choose any vertex  $v_i$  in the complete set of vertices ( $V$ ), setting

$$T_k = T_1 = (\{v_i\}, \phi).$$

**Step 2:** Find the edge of lowest cost ( $l'$ ) in  $L$  that connects any vertex of

$T_k$  to another vertex,  $v_j$ , belonging to  $V$  but not to  $T_k$ .

**Step 3:** Add  $v_j$  and  $l'$  to the tree  $T_k$ , creating a new tree  $T_{k+1}$ .

**Step 4:** Repeat Step 2 until all vertices have been included in the tree ( $T_n$ ).

Figure 2 shows the procedure for MST construction, showing the first three and the last iteration. If more than one edge of lowest cost could be inserted in step 2 of the algorithm, the MST would not be unique. In these cases, the MST would be dependent of the choice of the vertex  $v_i$  (Step 1) and of the order of evaluation of the links costs in the Step 2. However, as explained above, in the graphs associated to socio-economic data, this is unlikely to happen.

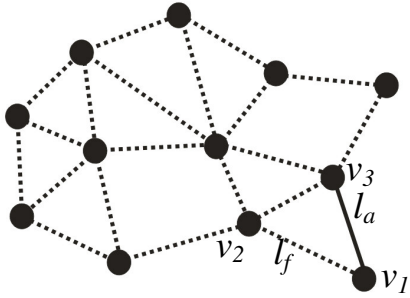
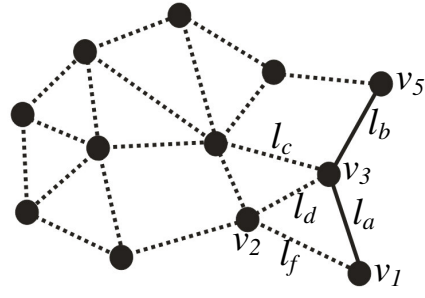
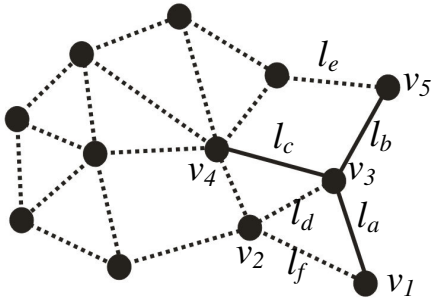
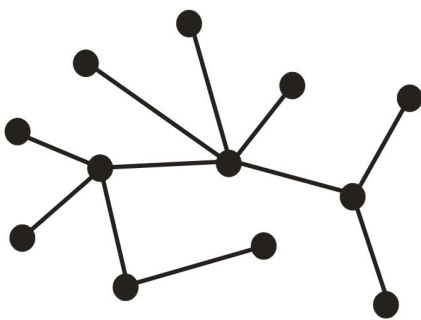
	<p><b>First iteration:</b></p> <p>Set <math>T_1 = (V_1, L_1)</math>, where <math>V_1 = \{v_1\}</math> and <math>L_1 = \emptyset</math>.</p> <p>Find the edge of lowest cost (<math>l_a \langle l_f</math>).</p> <p>Step 3: <math>T_2 \Rightarrow V_2 = \{v_1, v_3\}</math> e <math>L_2 = \{l_a\}</math>.</p> <p>Step 4: Repeat Step 2.</p>
	<p><b>Second iteration:</b></p> <p>Find the edge of lowest cost (<math>l_b \langle l_c \langle l_d \langle l_f</math>).</p> <p>Set <math>T_3 \Rightarrow V_3 = \{v_1, v_3, v_5\}</math> and <math>L_3 = \{l_a, l_b\}</math>.</p>
	<p><b>Third iteration:</b></p> <p>Find the edge of lowest cost (<math>l_c \langle l_d \langle l_e \langle l_f</math>).</p> <p>Set <math>T_4 \Rightarrow V_4 = \{v_1, v_3, v_4, v_5\}</math> and <math>L_3 = \{l_a, l_b, l_c\}</math>.</p>
	<p><b>Final Iteration:</b></p> <p><math>V_n = V</math>.</p>

Figure 2 - Construction of the minimum spanning tree.

### 3.4 MST partitioning

This section describes the partitioning algorithm for breaking up the MST into a set of contiguous clusters. Using the MST, we transform the regionalisation problem into a tree partitioning problem. To make a partition of  $n$  objects in  $k$  regions, it is necessary to remove  $k-1$  edges from the MST. Each resulting cluster will be a *tree*, with all vertices connected and no circuits (cf. Section 3.2). To make a partition of  $n$  objects in  $k$  trees, we use a *hierarchical division strategy*. Initially, all objects belong to a single tree. As we remove edges from the original MST, a *set of disconnected trees* appears, and each tree in this set has a univocal correspondence with a region. At each iteration, one of the trees is split into two by cutting out an edge, until we reach the number of clusters previously stipulated.

The partitioning algorithm produces a graph  $G^*$  that contains a set of trees  $T_1, \dots, T_n$  where each tree is connected, but has no common edges or vertices with the other trees. At the first iteration,  $G^*$  has only one tree, which is the MST. At each iteration, we examine the  $G^*$  graph, and take out one edge that will divide the tree  $T_i$  into two trees  $T_i^1$  and  $T_i^2$ . This is the same as splitting a connected region into two subregions. We select the edge that brings about the largest increase in the overall quality of the resulting clusters. The quality measure is the sum of the intracluster square deviations, which needs to be minimised:

$$Q(\Pi) = \sum_{i=0}^k SSD_i , \quad (2)$$

where:

- $\Pi$  is a partition of objects into  $k$  trees.

- $Q(\Pi)$  is a value associated to the quality of a  $\Pi$  partition.
- $SSD_i$  is the *sum of square deviations* in region  $i$ .

The intracluster square deviation SSD is a measure of dispersion of attribute values for the objects in a region. Homogeneous regions have small SSDs values. Thus, the smaller  $Q(\Pi)$ , the better the partition. The intracluster square deviation SSD is:

$$SSD_k = \sum_{j=1}^m \sum_{i=1}^{n_k} (x_{ij} - \bar{x}_j)^2 \quad (3)$$

where:

- $n_k$  is the number of spatial objects in tree  $k$ ;
- $x_{ij}$  is the  $j$ -th attribute of spatial object  $i$ ;
- $m$  is the number of attributes considered in analysis;
- $\bar{x}_j$  is the average value of  $j$ -th attribute for all objects in tree  $k$ .

At each iteration, we have to remove an edge from the graph  $G^*$  that contains a set of trees  $T_1, \dots, T_n$ . To do this, we compare the optimum solutions for each of the trees  $T_1, \dots, T_n$ . The solution that best subdivides a tree  $T$  is the optimum solution  $S_*^R$ , according to an objective function:

$$f_1(S_l^T) = SSD_T - (SSD_{T_a} + SSD_{T_b}) , \quad (4)$$

where:

- $S_l^T$  is the arrangement produced by cutting out the edge  $l$  from the tree  $T$ .
- $T_a$  and  $T_b$  are the two trees produced by dividing  $T$  after cutting out the edge  $l$ .

At each iteration, we divide the tree  $T_i$  that has the highest value of the objective function  $f_1(S_*^{T_i})$ . The idea is to get the greatest improvement of quality at each step. Starting from an MST, we produce the clusters as follows:

**Step 1:** Start the graph  $G^* = (T_0)$  where  $T_0 = \text{MST}$ .

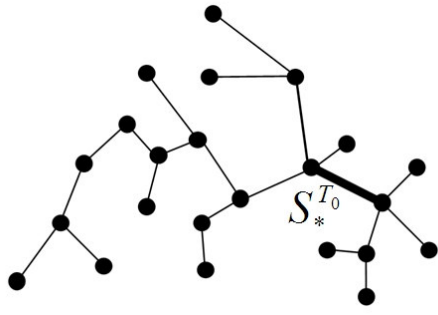
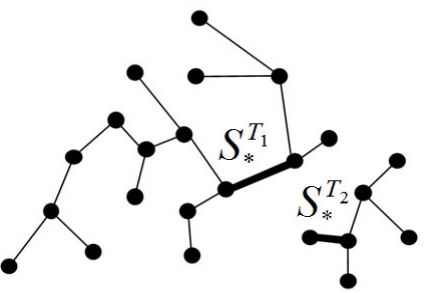
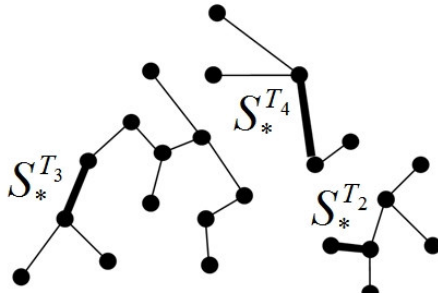
**Step 2:** Identify the edge that has the highest objective function  $S_*^{T_0}$ .

**Step 3:** While  $\#(G^*) < k$  (desired number of clusters), repeat steps 4 and 5.

**Step 4:** For all trees in  $G^*$ , select the tree  $T_i$  with the best objective function  $f_1(S_*^{T_i})$ .

**Step 5:** Split  $T_i$  into two new subtrees and update  $G^*$ .

Figure 3 pictures the method, showing the first three iterations. In the first iteration, there is only one tree in  $G^*$  (the MST). Then, we select the best objective function  $S_*^{T_0}$  and divide the tree by cutting out the edge matching  $S_*^{T_0}$ . This originates two new trees,  $T_1$  and  $T_2$ . By repeating the pruning, we will create an optimal partition of the MST. However, the exhaustive comparison of all possible values of the objective function is expensive computationally. To speed up the choice, we propose a heuristic described in detail in the next section.

 <p>The diagram shows a graph with 10 nodes and several edges. One edge, labeled <math>S_*^{T_0}</math>, is highlighted in bold black. This edge connects two nodes in the rightmost part of the graph.</p>	<p>Iteration 0: <math>G^* = \text{MST}</math>. We select the edge which has the largest objective function. Cut out this edge leaving two trees (<math>T_1</math> and <math>T_2</math>).</p>
 <p>The diagram shows the graph after the first iteration. The edge <math>S_*^{T_0}</math> has been removed. Two new edges are highlighted in bold black: <math>S_*^{T_1}</math> in the left part of the graph and <math>S_*^{T_2}</math> in the right part of the graph.</p>	<p>Iteration 1: <math>G^* = (T_1, T_2)</math>. We compare the highest objective functions for <math>T_1</math> and <math>T_2</math>. We split the tree <math>T_1</math> since <math>f_1(S_*^{T_2}) \leq f_1(S_*^{T_1})</math></p>
 <p>The diagram shows the graph after the second iteration. The edge <math>S_*^{T_1}</math> has been removed. Three new edges are highlighted in bold black: <math>S_*^{T_3}</math> in the left part, <math>S_*^{T_4}</math> in the middle part, and <math>S_*^{T_2}</math> in the right part.</p>	<p>Iteration 2: <math>G^* = (T_2, T_3, T_4)</math>. We compare the highest objective functions for <math>T_2</math>, <math>T_3</math> and <math>T_4</math>. We split the tree <math>T_3</math> since <math>f_1(S_*^{T_2}) \leq f_1(S_*^{T_4}) \leq f_1(S_*^{T_3})</math></p>
<p>Figure 3 – Partitioning of the MST</p>	

### 3.5 A heuristic for fast tree partitioning

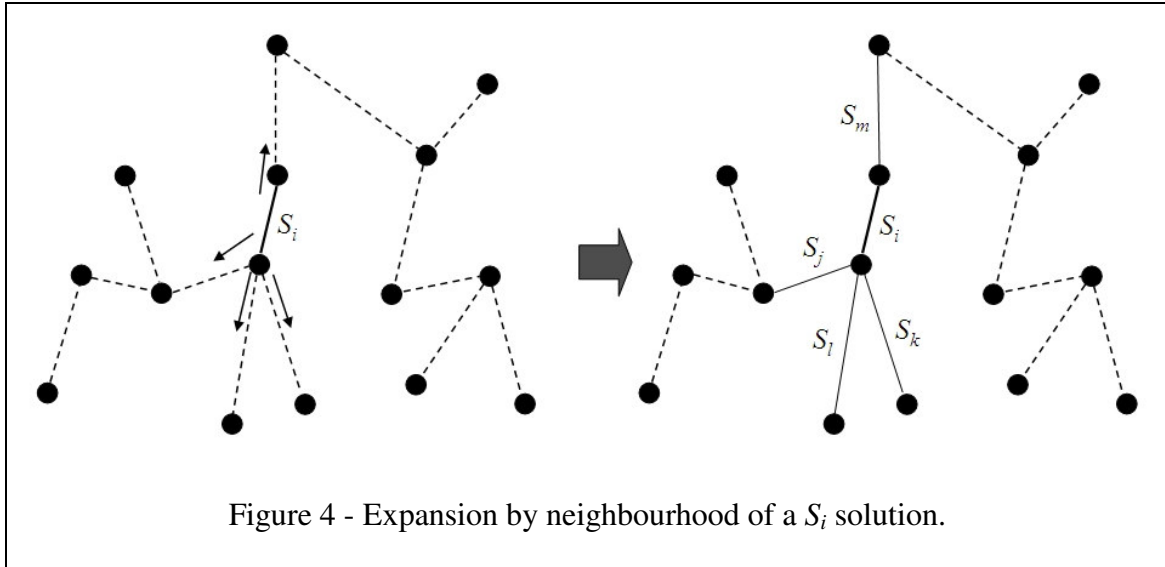
In this section, we describe a heuristic procedure that speeds up MST partitioning. In section 3.4 above, we described the MST partitioning algorithm. We pointed out that, for each subtree, we need to find the edge that best subdivides it. At each iteration, the algorithm chooses the edge that maximises the objective function (equation 4). However, we made no mention of the computational demands involved in choosing the best solution. In fact, choosing the best edge to partition each subtree is computationally demanding. In the exhaustive case, we would need to examine all the arrangements leading to  $k$  clusters, and select the edge that leads to the optimal solution. This is an NP-hard problem that leads to a combinational explosion. Therefore, this section describes an efficient heuristic that approximates the optimal solution at acceptable speeds.

Taken as an optimisation problem, the search for the edge that best subdivides a tree equals the search for the best candidate within a solution space  $S$ :

$$S = \{S_1, S_2, \dots, S_{n-1}\} \quad , \quad (5)$$

where the  $S_l$  candidate is the removal of edge  $l$  from the tree. The algorithm searches for an optimum solution by analyzing the neighbours of candidates already visited. We start by evaluating the solution  $S_i$  at the current vertex and the solutions in its neighbourhood. Figure 4 shows how we do the expansion by neighbourhood in the solution space. In the example,  $S_i$  has four neighbours:  $S_j$ ,  $S_k$ ,  $S_l$  and  $S_m$ . We evaluate all five solutions, and keep track of the solution with the highest objective function  $f_l(S_i)$  (cf. Equation 2).





After finding out the best solution in a given neighbourhood, the next step is to select a vertex for expanding our search for even better solutions. Contrary to common sense, the proper choice is not necessarily the vertex with the highest value of objective function. The main problem with all optimisation techniques is to avoid choosing solutions that are local maximums, instead of the desired global maximum, by also examining candidates that do not bring an immediate improvement to the objective function. To do this, we use a second objective function  $f_2$ , which selects solutions that will divide the tree into two groups that are more homogeneous and balanced. The  $f_2$  function prevents the generation of subtrees that are very uneven in size. Its value for a vertex is the smaller of two differences between the SSD value of the current tree and SSD values of the two *subtrees* resulting from cutting out this vertex:

$$f_2 = \min[(SSD_T - SSD_{T_a}), (SSD_T - SSD_{T_b})] \quad . \quad (6)$$

A good choice for the starting point can reduce the iterations needed for the search strategy to find the optimum solution. Considering the character of *hierarchical division*

methods, a satisfactory starting point is a vertex located in the centre of the tree. To find out the central vertex, we go over all edges until we identify the edge that best splits the tree into two *subtrees* of similar sizes. Then we continue with the optimisation using the two objective functions. The stopping condition (*SC*) of the exploration strategy is the maximum number of iterations without growth in the value of  $f_l$ . In summary, the optimisation heuristic is:

**Step 1:** Start from the central vertex,  $V_c$ . Insert solutions associated to edges incident in  $V_c$  in the list of potential solutions,  $S_p$ . Set  $n^* = n = 0$  and  $f_l(S^*) = 0$ .

**Step 2:** Evaluate solutions in  $S_p$  and store them in the list  $L$ .

**Step 3:** Update the information on best available solution. Select the solution  $S_j$  in  $S_p$  with the highest objective function  $f_l(S_j)$ . If  $f_l(S_j) > f_l(S^*)$ , then  $S^* = S_j$  and  $n^* = n$ .

**Step 4:** Set  $n = n + 1$ . Based on the balancing function  $f_2(S_j)$ , select in the list  $L$  a solution that will have its neighbourhood expanded, originating a new list of potential solutions  $S_p$ .

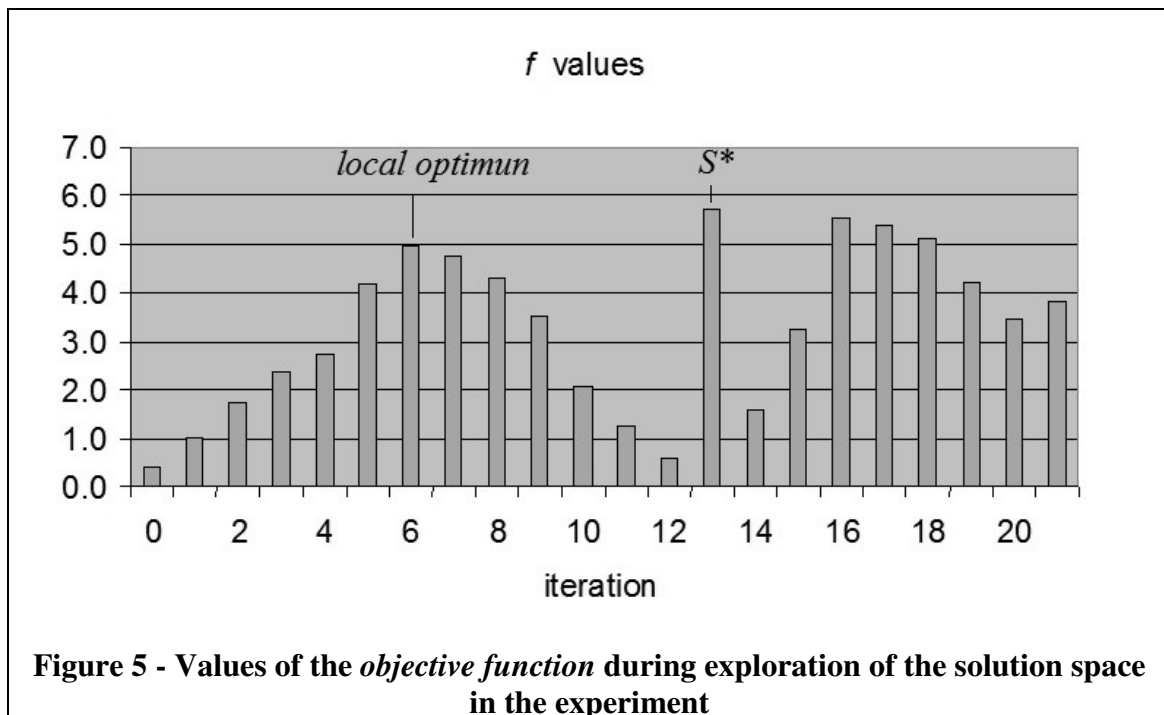
**Step 5:** Check the stopping condition ( $n - n^* > SC$ ). If it is false, go back to *Step 2*. Otherwise, choose  $S^*$  as the best available solution and finish.

In this procedure, we take:

- $S^*$  is the best interim solution, which on completion will represent the chosen solution;
- $n$  is the number of iterations;
- $n^*$  is the last iteration where an improvement of the objective function  $f_l$  has occurred;

- $S_p$  is the list of potential solutions in the current iteration;
- $L$  is the list of candidates that have been evaluated but not yet expanded;
- $S_j$  is the best solution in the current list of potential solutions  $S_p$ . We identify this solution after considering all solutions in  $S_p$ .
- $SC$  is the stopping condition for the search, defined as the number of iterations without improvements in  $f_l$ .

To show the behaviour of the heuristic, we present one experiment where we chose a vertex far from the best solution as the starting point. Figure 5 presents a bar chart with the values of the  $f_l$  objective function. We can note a local maximum, which was reached in the seventh iteration. To go beyond it, the search strategy had to insist along six iterations before the value of the objective function started to increase again. We found the optimum solution at the fourteenth iteration. In this experiment, the stopping condition was eight iterations without improvement in the  $f_l$  objective function.



In the experiment, an MST vertex far from the ideal solution was the seed of the exploration procedure. With this choice, we intended to show how the search strategy escapes from local maximums. In a real case, we should select a starting point that reduces the iterations needed to find an optimum solution. Considering the character of *hierarchal division* methods, a good starting point is a vertex located in the centre of the tree. Our tests show that adopting a central vertex as starting point reduces the number of evaluations to find the best solution.

### 3.6 Performance impact of the heuristic

This section discusses the performance impact of the proposed heuristic, compared with the exhaustive search method. The data set has 415 municipalities in the Brazilian state of Bahia, which were grouped in 20 regions based on 3 attributes. These attributes measure the percentage of municipality area with 3 land cover types: crops, pasture and woods. We used two different values for the stopping condition ( $SC = 15$  and  $SC = 30$ ). Higher values of the stopping condition  $SC$  result in the better quality partitions, at a higher computational cost. Table 1 presents results, comparing the exhaustive search (where no heuristic is applied) with the heuristic using two different stopping conditions.

**Table 1- Performance comparison of MST partitioning methods**

	<b>Exhaustive search</b>	<b>Heuristic with SC = 30</b>	<b>Heuristic with SC = 15</b>
<b>Q(<math>\Pi</math>) – partition quality</b>	380.22	380.22	390.88
<b>Number of evaluations</b>	8060	1818	1121
<b>Relative performance (no optimisation = 100)</b>	100	31	17

Table 1 compares the quality of the resulting partitions, the number of evaluations and the relative performance. The quality measure is the sum of the

intracluster square deviations, which needs to be minimised (equation 2). The exhaustive search produces the best quality, at a large performance cost. The heuristic procedure with  $SC=30$  also achieves an optimal quality, at 30% of the running time of the exhaustive search. The heuristic with  $SC=15$  approximates the optimal quality at 17% of the running time. The execution time in Table 2 does not include time spent for MST generation, which is the same for all three cases. We also provide the number of evaluations performed by each method. An evaluation is the set of all operations in one loop of the MST partitioning algorithm described in Section 3.4. Each evaluation selects one possible edge to be removed.

#### **4 Applying SKATER: a case study in São Paulo**

In this section, we show how to apply the SKATER algorithm for a case study in city of São Paulo, Brazil. The study has two parts: the first study shows results without restrictions in the regionalisation procedure. The second part shows how SKATER can incorporate restrictions. The data for the study consist of the “Social Exclusion/Inclusion Map of the City of São Paulo” (Câmara et al., 2004). Based on data from the 1991 census and the legal division of São Paulo into 96 districts, the social exclusion/inclusion map has four socio-economic indexes: income distribution, quality of life, human development and gender equality. The indexes are normalised in a continuous scale of  $[-1, 1]$ .

The first study consists in getting eight homogenous regions for São Paulo, starting from the 96 districts and using the four socio-economic indicators of the Social Exclusion/Inclusion Map. We used the SKATER algorithm as described in the previous section. Figure 1.a shows the districts map and the connectivity graph and Figure 1.b

shows the MST. The tree partitioning requires cutting out seven edges (Figure 6.a). The result of the unrestricted regionalisation (shown in Figure 6.a) has great variations in the number of districts of each region. The northeastern region has one district only and has 12,408 inhabitants, which is 0.12% of total population in the municipality. The bigger region next to it has 36 districts with a population of 3,533,082 inhabitants, which is 36.6% of total population. For many applications, this unbalance is not convenient. Therefore, we need to add restrictions to the regionalisation procedure. In studies involving rare events, for example, it may be necessary to establish minimum values for the population at risk in the resulting clusters, for resulting rates to be representative.

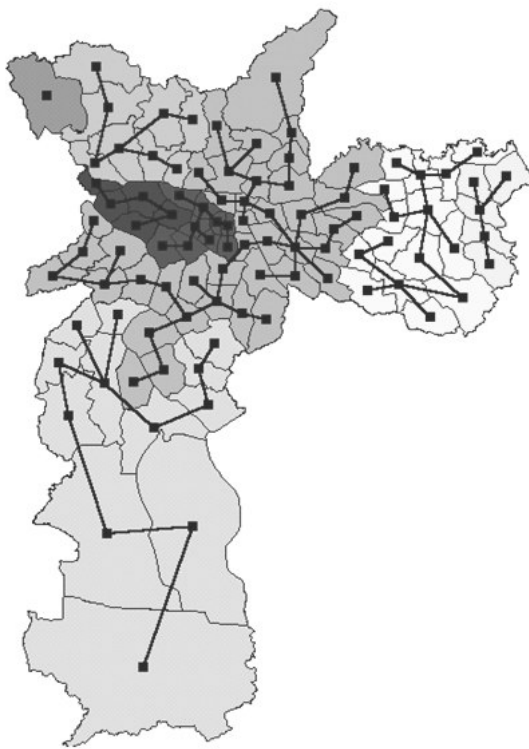


Figure 6.a - Unrestricted regionalisation.

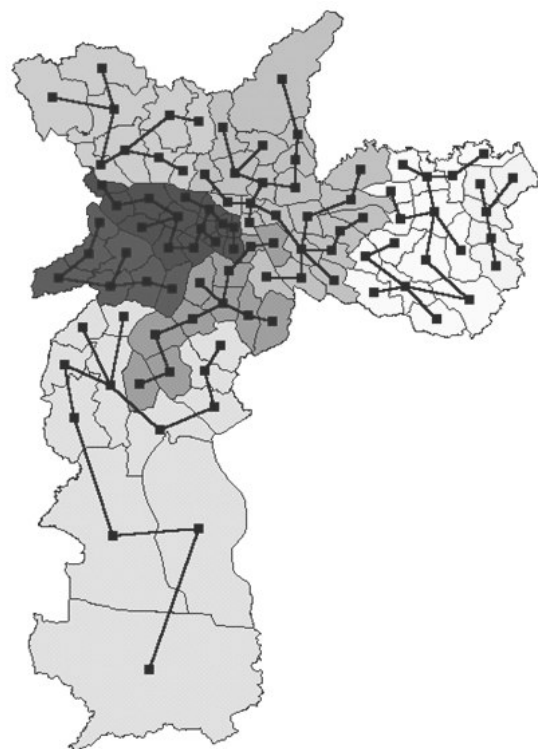


Figure 6.b - Population-restricted regionalisation

A simple way to add restrictions to SKATER is to set upper and lower limits for certain attributes of a region. If a solution is off-limits, it will be considered invalid. Thus, the search strategy includes both a condition of homogeneity of regions and a

condition for minimum or maximum value of attributes in a region. We performed a second regionalisation procedure with a constraint of a minimum population of 500,000 inhabitants for a region. The result is shown in Figure 6.b. Regions are more balanced in population, which now varies from 651,513 to 2,543,743 inhabitants.

## **5 Comparison between SKATER and AZP**

In this section, we compare the SKATER and the AZP algorithms. We selected AZP since it is a well-known algorithm and has been used for regionalisation of the UK Census (Openshaw and Rao, 1995, Openshaw and Alvanides, 2001, Alvanides et al., 2002, Martin, 2003). The AZP algorithm is also a constrained classification procedure of the same type as SKATER. We recognise there is no decisive way to compare two classification methods, because performance is dependent of characteristics of the data set used in the study. Nevertheless, we can detect some trends with well-selected experiments. We compared SKATER with AZP in a series of experiments with different arrangements of number of objects, number of attributes, and resulting regions. The grouping condition used by the two algorithms in the two experiments was the internal homogeneity of the regions. We used three data sets in the comparison: a) 96 districts of the municipality of São Paulo; b) 415 municipalities of the Brazilian state of Bahia state; and c) 853 municipalities of the state of Minas Gerais. The attributes used in the experiments were socio-economic variables.

We implemented the AZP as described in Openshaw and Rao (1995). We used the same stopping condition for both methods in all experiments ( $SC = 10$ ). In our tests, the AZP method has shown to be sensitive to the choice of the initial partition, changing the quality of resulting partitions in a significant way. This fact suggests that this

method has a tendency to converge to local maximums. This problem has been pointed out in Openshaw and Rao (1995). Following their suggestion, we ran the AZP method 5 times for each experiment, using different initial partitions. We took the best value of the 5 runs as the partition quality for the AZP method.

We compared SKATER and AZP for processing speed and quality of the resulting partitions. We tested three cases: different number of objects, different number of attributes, and different number of resulting regions. The first experiment used the three data sets that have a different number of objects (96 for São Paulo, 415 for Bahia, and 845 for Minas Gerais). The number of resulting clusters and the number of attributes were the same (20 clusters and 3 attributes). The results are shown in Table 2. The second experiment used the same data set (415 municipalities of state of Bahia), the same number of resulting regions (20) and varied the number of attributes (3, 6, and 9 attributes). The results are shown in Table 3. The third experiment used the same data set (415 municipalities of Bahia state), the same number of number of attributes (3) and varied the number of resulting clusters (20, 40, 60). The results are shown in Table 4.

**Table 2 - Comparison of AZP and SKATER for different numbers of objects**

Data sets: (a) 96 districts of the city of São Paulo; (b) 415 municipalities of the state of Bahia; (c) 845 municipalities of the state of Minas Gerais. All tests used 3 attributes and 20 regions.

num. Objects	Skater		AZP		Q(P)%	time%
	Q(P)	Time (s)	Q(P)	Time (s)		
<b>96</b>	5.046	10	7.0584	92.2	0.715	0.108
<b>415</b>	403.32	216	408.072	2588.6	0.988	0.083
<b>853</b>	3.141	889	3.1874	9206	0.985	0.097



**Table 3- Comparison of AZP and SKATER for different numbers of attributes**

Data set: 415 municipalities of the state of Bahia with 20 clusters

num. attributes	Skater		AZP		Q(P)%	time%
	Q(P)	Time (s)	Q(P)	Time (s)		
<b>3</b>	403.3	216	408.1	2588.6	0.988	0.083
<b>6</b>	806.6	225	869.8	3904.4	0.927	0.058
<b>9</b>	1210.0	234	1324.4	6011.0	0.914	0.039

**Table 4 - Comparison of AZP and SKATER for different numbers of clusters**

Data set: 415 municipalities of the state of Bahia with 3 attributes

num. Clusters	Skater		AZP		Q(P)%	time%
	Q(P)	Time (s)	Q(P)	Time (s)		
<b>20</b>	403.3	216	408.1	2588.6	0.988	0.083
<b>40</b>	245.7	256	314.0	4316	0.782	0.059
<b>60</b>	172.9	274	254.9	7988	0.678	0.034

The criterion for quality comparison was the homogeneity of the resulting regions, measured by Equation 2 (see above), where a smaller value is better. SKATER was superior to AZP in all tests. The relation among the partition quality values  $[Q(\Pi_{AZP})/Q(\Pi_{SKATER})]$  varied between 0.678 a 0.988, and the average value was 0.856. The processing speed of SKATER was much superior to AZP, ranging from to a gain of 10 to almost 30 times. The relative gain in processing speed of SKATER over AZP improved as the number of attributes increased (Table 3). The relative advantage in

performance of SKATER over AZP also increased with the number of resulting clusters (Table 4).

## 6 Conclusion

In this paper we present SKATER (*Spatial 'K'luster Analysis by Tree Edge Removal*), an efficient method for regionalisation of socio-economic units represented as spatial objects, which combines the use of a minimum spanning tree with combinational optimisation techniques. The main algorithm has a fast convergence towards a best-cost solution. The heuristic optimisation techniques used in the tree partitioning step allow significant speedup without quality degradation. In comparison with the well-known AZP regionalisation method, SKATER produces good quality results and is significantly faster. The algorithm also allows including restrictions in the regionalisation procedure. In cases where homogeneity of regions is an important need, it produces a good quality partition in a short time. We consider SKATER to be a good choice for regionalisation, especially when applied to large data sets. The algorithm is available as an open source software, as part of the open source GIS library TerraLib (<http://www.terralib.org>), and is included in the TerraView visualisation and analysis GIS (<http://www.terralib.org/terraview>).

## Acknowledgments

Renato Assunção's research is partially supported by CNPq (grant 305567/2004-7). Marcos Neves' research has been supported by EMBRAPA. Gilberto Camara's work is partially funded by CNPq (grants PQ - 300557/19996-5 and 550250/2005-0) and FAPESP (grant 04/11012-0). Corina Freitas' research has been partially supported by CNPq (grant 305546/2003-1).

## References

- AHO, A. V., HOPCROFT, J. E. and ULLMAN, J. D., 1983, *Data structures and algorithms*. (Reading, Mass.: Addison-Wesley).
- ALVANIDES, S., OPENSHAW, S. and REES, P., 2002, Designing your own geographies. In *The Census Data System*, P. Rees, D. Martin and P. Williamson (Ed.) (Chichester: Wiley), pp. 47-65.
- CÂMARA, G., SPOSATI, A., KOGA, D., MONTEIRO, A. M., RAMOS, F., DRUCK, S. and CAMARGO, E., 2004, Mapping Social Exclusion/Inclusion in Developing Countries: Social Dynamics of São Paulo in the 90's. In *Spatially Integrated Social Science: Examples in Best Practice*, M. Goodchild and D. Janelle (Ed.) (London: Oxford University Press).
- EVEN, G., NAOR, J., RAO, S. and SCHIEBER, B., 1997, Fast approximate graph partitioning algorithms. In *Proceedings of the 8th ACM-SIAM symposium on discrete algorithms*, (New Orleans, Louisiana, United States). pp. 639 - 648.
- HAINING, R., WISE, S. and MA, J., 2000, Designing and implementing software for spatial statistical analysis in a GIS environment. *Journal of Geographical Systems*, **2**, 257-286.
- JUNGNICKEL, D., 1999, *Graphs, Networks and Algorithms*. (Berlin: Springer).
- KARYPIS, G. and KUMAR, V., 1998 Multilevel algorithms for multi-constraint graph partitioning. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, (San Jose, CA).
- MARTIN, D., 1998, Optimizing census geography: the separation of collection and output geographies. *International Journal of Geographical Information Science*, **12**, 673-685.
- MARTIN, D., 2003, Extending the automated zoning procedure to reconcile incompatible zoning systems. *International Journal of Geographical Information Science*, **17**, 181-196.

- OPENSHAW, S., 1977, A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Transactions of the Institute of British Geographers (New Series)*, **2**, 459-472.
- OPENSHAW, S. and ALVANIDES, S., 1999, Zone design for planning and policy analysis. In *Geographical Information and Planning*, J. Stillwell, S. Geertman and S. Openshaw (Ed.) (Berlin: Springer), pp. 299-315.
- OPENSHAW, S. and ALVANIDES, S., 2001, Designing zoning systems for representation of socio-economic data. In *Time and Motion of Socio-Economic Units*, A. Frank, J. Raper and J. Cheylan (Ed.) (London: Taylor and Francis).
- OPENSHAW, S., ALVANIDES, S. and WHALLEY, S., 1998, Some further experiments with designing output areas for the 2001 UK census. Report (Leeds: University of Leeds).
- OPENSHAW, S. and RAO, L., 1995, Algorithms for reengineering 1991 Census Geography. *Environment and Planning A*, 425-446.
- OPENSHAW, S. and WYMER, C., 1995, Classifying and regionalizing census data. In *Census Users' Handbook*, S. Openshaw (Ed.) (Cambridge, UK: Geoinformation International), pp. 239-269.

