J Geograph Syst (2001) 3:167–180

Journal of Geographical Systems
© Springer-Verlag 2001

Redistricting in a GIS environment: An optimisation algorithm using switching-points

W. Macmillan

School of Geography, University of Oxford, Mansfield Road, Oxford OX1 3TB, UK (e-mail: bill.macmillan@geog.ox.ac.uk)

Received: 25 July 2000 / Accepted: 1 February 2001

Abstract. This paper gives details of an algorithm whose purpose is to partition a set of populated zones into contiguous regions in order to minimise the difference in population size between the regions. The algorithm, known as SARA, uses simulated annealing and a new method for checking the contiguity of regions. It is the latter which allows the algorithm to be used to tackle large problems with modest computing resources. The paper describes the new contiguity checking procedure, based on the concept of switching points, and compares it with the connectivity method developed by Openshaw and Rao [1]. It goes on to give a detailed description of the algorithm, then concludes with a brief discussion of possible extensions to accommodate additional zone-design criteria.

1 Introduction

The origins of this work lie in a project on political redistricting but the algorithm described below is capable of being used in connection with a wide range of zone design problems. The standard problem in political redistricting is to partition a set of contiguous, populated zones into a smaller number of contiguous regions, in such a way that the sizes of the regional populations deviate minimally from the average size. Any problem that is capable of being cast in this form is amenable to solution by version 1 of the Simulated Annealing Redistricting Algorithm (SARA). As one would expect with a simulated annealing procedure, the algorithm cannot guarantee that a global optimum has been found but it can give the user a high degree of confidence that the terminal solution is close to optimal. Moreover, as the complexity of the design problem increases, SARA is likely to do considerably better than informal, manual approaches.

A variety of different procedures have been suggested for computer-assisted zone design, many of which seek solutions which are, at best, locally optimal (see [2] and [3]). The use of simulated annealing to find global optima was first suggested by Bowdry [4]. This approach was tested against other optimisation procedures by Pierce

The author is grateful to Seraphim Alvanides of the School of Geography, University of Leeds for his contribution to the development work associated with this paper.

[2] and was found to perform relatively well. However, Pierce demonstrated that, without an efficient contiguity checker or massive computing power, it is suitable for small problems only. The contiguity checker in SARA was devised to circumvent this difficulty.

SARA is an algorithm which pursues optimality whilst preserving contiguity. It operates on a map consisting of nodes and arcs, which, together, constitute the boundaries for a set of contiguous zones. Each zone is characterised by a label (the zone's identifier) and a number that represents it population (or some similar characteristic of the zone). The algorithm requires an initial partition to be selected. A partition consists of an allocation of zones to regions with the following characteristics: each zone is allocated to one and only one region; each region has at least one zone; and the zones in each region are contiguous. Any such configuration will serve as the initial partition. The pursuit of optimality involves the alteration of the partition, one stage at a time, with each stage involving the movement of a single zone from one region to another. The move takes place only if the contiguity of the partition is preserved.

Various functions could be used to measure the deviation of the regional populations from the average or target population. The sum of the squares of the deviations is one obvious possibility. Another is the sum of their absolute values. SARA uses the latter criterion. A contiguity-preserving move of a zone from one region to another is accepted if it reduces the combined population deviation of the donor and recipient regions. However, it is also accepted, with a certain probability, if the deviation is increased. This is an essential feature of simulated annealing. The reason for adopting such a strategy derives from the problem's solution space. It is a tree-like structure (see [5]). If the algorithm only accepted moves that improved the objective function, the search procedure would have a tendency to become trapped in branches with local but not global optima. The possibility of accepting moves that entail a short-term deterioration in the objective function opens up the possibility of escaping from dead ends in the solution space.

The likelihood of accepting a deviation-increasing move is determined by the size of the increase and a parameter that is referred to in simulated annealing as the 'temperature' (by analogy with the temperature of a metal undergoing the process of annealing). The higher the value of this parameter, the more likely it is that a move will be accepted, all else being equal. As the search for the solution progresses, the temperature parameter is reduced, making it less and less likely that adverse changes will be accepted. Each temperature is maintained until threshold numbers of successful and unsuccessful zone transfers have been exceeded.

In outline, the algorithm proceeds as follows:

Initialisation: Choose an initial partition and an initial value for the 'temperature' parameter; set the counters for the number of successful and unsuccessful swaps to zero.

Step 1: Select an over-populated region for the removal of a zone (do not select a region with one zone only).
Step 2: Choose the zone to be removed from the donor region.
Step 3: If contiguity would be lost by the transfer of this zone return to Step 2.
Step 4: Select a recipient region for the chosen zone from amongst the regions to which neighbouring zones belong (do not choose the donor region).
Step 5: (a) if the transfer would decrease the combined population deviation of the donor and recipient regions then accept it;

(b) if the transfer would increase the combined population deviation of the donor and recipient regions then accept it with a probability governed by the size of the deviation and the value of the temperature parameter.

Step 6: If the transfer is accepted, calculate the new regional population deviations and add one to the count of successful transfers; if it is not accepted, add one to the count of unsuccessful transfers.

Step 7: If the aggregate regional population deviation is within the target range then stop; if the threshold numbers of successful and unsuccessful swaps have not been exceeded then go to Step 1; if the thresholds have been exceeded then reduce the value of the temperature parameter then go to Step 1.

The part of the algorithm that poses the greatest computational challenge is the contiguity check in Step 3. The connectivity approach to this problem is described in Sect. 2. The alternative method used in SARA, based on so-called switching points, is detailed in Sect. 3. In Sect. 4, the performance of the two methods is compared.

2 Contiguity checking using the connectivity method

The method of contiguity checking employed by Openshaw and Rao [1] focuses on a region – the region that would remain if the target zone was removed from it. The new method, proposed in outline in [5], focuses on a single zone – the target zone itself. At the heart of the Openshaw and Rao method lies the connectivity of the map. It is possible to represent map topology using a connectivity matrix with terms c_{ij}, where $c_{ij} = 1$ if zones i and j have a common border and $c_{ij} = 0$ otherwise. Thus, the top left hand corner of the connectivity matrix for the map in Fig. 1 is as shown in Table 1.

For any subset of zones, there is a corresponding submatrix. The submatrix for the light region in Fig. 1 is given in Table 2. To establish the connectivity of a region, it is sufficient to power the regional submatrix (multiply it by itself) up to $n-1$ times, where n is the number of zones in the region. If and when all elements of the resultant matrix become non-zero, the region has been shown to be contiguous. If some elements remain zero in the $(n-1)$-th powered matrix, then the region is not contiguous. This method for establishing contiguity is neat mathematically (it is easily stated algebraically and produces ancillary information about the degrees of connectivity of zones to each other) but it is inefficient computationally.

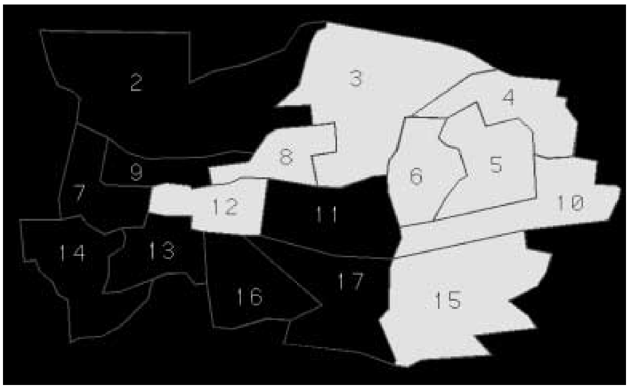

Fig. 1. Base map

Table 1. Connectivity matrix for the map in Fig. 1 (numbers in margins are zone identifiers, zone 1 being the exterior zone)

	1	2	3	4	5	…
1	1	1	1	1	0	…
2	1	1	1	0	0	…
3	1	1	1	1	0	…
4	1	0	1	1	1	…
5	0	0	0	1	1	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 2. Connectivity sub-matrix for the light region in Fig. 1 (numbers in margins are zone identifiers)

	3	4	5	6	8	10	12	15
3	1	1	0	1	1	0	0	0
4	1	1	1	1	0	1	0	0
5	0	1	1	1	0	1	0	0
6	1	1	1	1	0	1	0	0
8	1	0	0	0	1	0	1	0
10	0	1	1	1	0	1	0	1
12	0	0	0	0	1	0	1	0
15	0	0	0	0	0	1	0	1

The strategy behind the Openshaw and Rao algorithm may be thought of as a streamlined method for establishing contiguity using the connectivity matrix. It begins, in effect, with a list of the zones in the region and proceeds by removing from this list those zones that are shown to be connected, directly or indirectly, to an arbitrarily chosen starting zone. In the example shown in Fig. 2 and Tables 3 and 4, the search starts with zone 3 (which is indicated by the assignment of the largest dot to this zone) and removes the zones it is connected to, namely 4, 6 and 8 (which have the

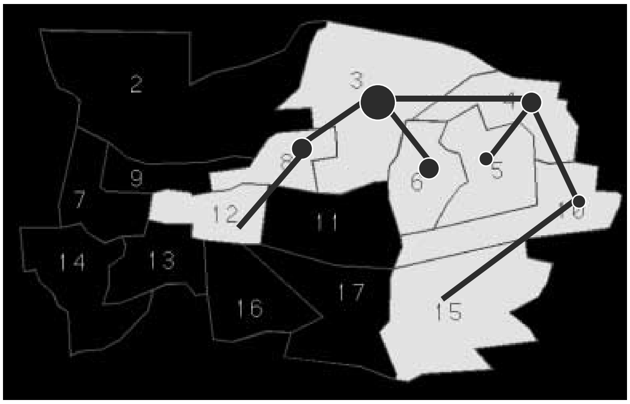

Fig. 2. An illustration of the Openshaw and Rao method for establishing that the light-shaded region is contiguous

Table 3. Connectivity search through the sub-matrix for the light region in Fig. 1

	3	4	5	6	8	10	12	15
3	1	1	0	1	1	0	0	0
4	1	1	1	1	0	1	0	0
5	0	1	1	1	0	1	0	0
6	1	1	1	1	0	1	0	0
8	1	0	0	0	1	0	1	0
10	0	1	1	1	0	1	0	1
12	0	0	0	0	1	0	1	0
15	0	0	0	0	0	1	0	1

Table 4. The connectivity search from Table 3 using zone lists

From	To	Zone list
		3,4,5,6,8,10,12,15
3	4,6,8	5,10,12,15
4	(3),5,(6),10	12,15
5	(4),(6),(10)	12,15
10	(4),(5),(6),15	12
6	(3),(4),(5),(10)	12
8	(3),12	Contiguity established

next-largest dots). The first of these is picked for the next stage of the search and so on, with backtracking whenever the search draws a blank (see Table 4 and the arrows on the matrix in Table 3). In fact, the Openshaw and Rao algorithm does not make any explicit use of the connectivity matrix. It operates directly on lists of zone neighbours but these lists are, of course, the identifiers of the zones for which ones appear in the rows of the connectivity matrix, excluding the ones on the diagonal.

To check whether contiguity would be lost by the removal of a zone, the algorithm operates on the submatrix (sublist) consisting of the zones that would remain in the region. If zone 12 was the target for removal, the procedure would be as in Table 4 but with 12 deleted throughout (see Fig. 3). If zone 8 was the target zone, the procedure would be as in Table 5 (see Fig. 4).

The performance of the Openshaw and Rao algorithm is acceptable for small problems with adequate computing resources. For example, in a test problem in which 930 zones were divided into 5 regions, the solution process required 650,000 DO loops, 9.5 million assignments, and 14 million IF statements for just one run. The performance of the algorithm varies with the number of zones and regions and is dependent on the form of the map. Moreover, when used as part of a simulated annealing procedure, computing times become lengthy even for small problems.

3 The switching point method

As noted above, the switching point method focuses on the zone that is targeted for removal from a region rather than on the region as a whole. It relies on the topological observation that loss of contiguity entails the creation of two or more new sections of 'external' boundary in the separated parts of the region.

Table 5. Procedure for checking contiguity with zone 8 removed

From	To	Zone list
		3,4,5,6,10,12,15
3	4,6	5,10,12,15
4	(3),5,(6),10	12,15
5	(4),(6),(10)	12,15
10	(4),(5),(6),15	12
6	(3),(4),(5),(10)	12
End		Discontiguity established

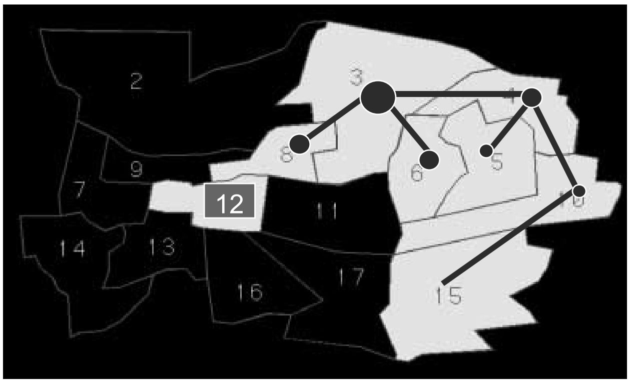

Fig. 3. An illustration of the Openshaw and Rao method for establishing that contiguity would not be lost by the removal of zone 12

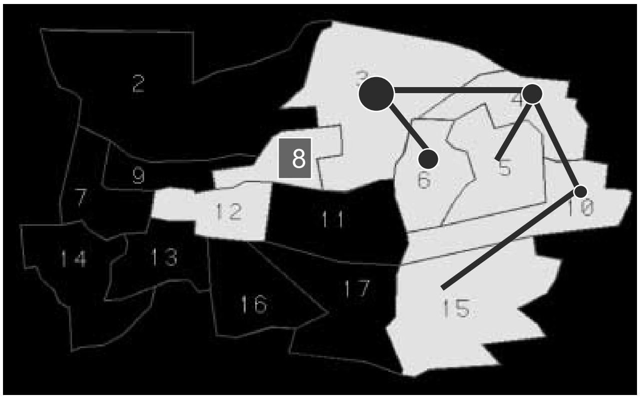

Fig. 4. An illustration of the Openshaw and Rao method for establishing that contiguity would be lost by the removal of zone 8

To understand the method, it is important to be clear about the distinction between internal and external boundaries. Every arc in the map is a boundary between two zones. If both zones currently belong to the same region, the boundary is said to be

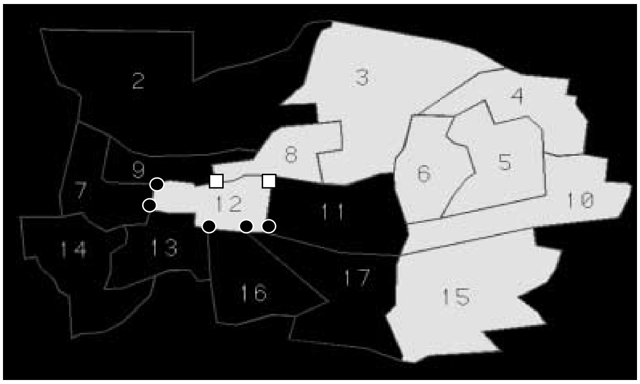

Fig. 5. Boundary nodes for zone 12 with the 2 switching points highlighted as white squares

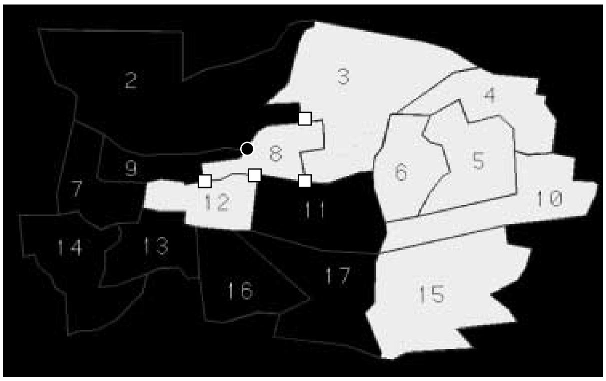

Fig. 6. Boundary nodes for zone 12 with the 4 switching points highlighted as white squares

internal to the region. If the zones currently belong to different regions, the boundary is said to be external – it is part of the edge of the region as a whole. For every zone, each of the arcs which make up its boundary are, with respect to the zone's current region, either internal or external boundary segments. Thus, in circulating around the boundary of the zone from one arc to the next, the arc type can switch (from internal to external or vice versa) or it can stay the same (both arcs are internal or both are external). A node (arc endpoint) at which a switch takes place is called a switching point. Thus, each new section of external boundary that would be created by the removal of a zone from its current region has two switching points, one at each end. The algorithm establishes whether a zone may be transferred from one region to another by determining the number of switching points it posses in the current partition.

Consider, for example, the possibility of removing zone 12 from the light region. Only one new section of external boundary would result – the boundary between zones 8 and 12 (see Fig. 5). However, if zone 8 is removed, two new external boundary sections would be created, the 8–12 boundary and the 8–3 boundary (see Fig. 6). By counting the number of switching points, it is possible to establish the number of new external boundary sections that would be created in each case. Imagine taking a tour round the boundary of zone 12 in Fig. 5, starting at the node shared by 7, 9 and 12.

Moving to the right, the 9–12 boundary is external to zone 12's current region. At the next node (shared by 8, 9 and 12), the boundary switches to being internal to the region so this node is a switching point. The following node is also a switching point as the boundary changes back to being external. For the remainder of the tour, the boundary remains external so there are no other switching points. Thus, the total switching point count is two indicating that only one new section of external boundary would be created by the removal of zone 12 so contiguity would not be lost. The removal of 12 would take a bite out of the region but would not bite through it. Now imagine taking a tour round zone 8 (Fig. 6). Four of the nodes are switching points, indicating that two new external boundary sections would be created so that contiguity would be lost. The removal of 8 would bite right through the region.

The above illustration has shown that there can be two or four switching points. In general, there can be zero or $2n$, where n is a positive integer representing the number of new sections of external boundary. If the count is six, three new sections of external boundary would be created by the zone's removal. For example, if zones 7 and 16 were in the light region, the number of switching points for zone 12 would rise to six; its removal would create a three-part region – zone 7 on its own, zone 16 on its own and the rest of the zones together. If the count is zero, there are two possible reasons: (1) the zone is completely surrounded by zones belonging to the *same* region so it would not be possible to attach it to another region without creating a discontiguity (this applies to zone 5); (2) the zone is completely surrounded by zones of *other* regions so its removal would leave its region empty, contrary to the requirement that each region should have at least one zone. In either case, the removal of the zone should not be allowed.

As with the Openshaw and Rao algorithm, the switching point method uses zone lists but they do not correspond directly to the lists produced by writing out the identifiers of the non-zero terms in the connectivity matrix, reading from left to right. Rather, the lists are compiled by making a tour of the zone boundary starting from an arbitrarily chosen node. The resultant data is held in an adjacent zone table and the algorithm adjusts a corresponding adjacent region table. An adjacent zone table for the example problem is shown on the left hand side of Table 6 (the tables are not unique). An adjacent region table is shown on the right. This table replicates the adjacent zone table except that the zone identifier is replaced by the identifier of the region to which the zone currently belongs: region 1 is the external region; region 2 the light region; and region 3 the dark region.

To establish whether or not a target zone may be removed from a region without destroying its contiguity, the switching point count is made using changes in the body of the adjacent region table from same-region to different-region labels and *vice versa*. Table 7 reproduces the adjacent region table entries for zones 8 and 12 and shows the corresponding counts of switching points (the switching point line shows the running

Table 6. An adjacent zone table (data) and a corresponding adjacent region table (variables) for the map in Fig. 1

Zone	Adjacent zones (data)	Region	Adj. regions (variables)
2	1,3,8,9,7,1	3	1,2,2,3,3,1
3	1,4,6,11,8,2,1	2	1,2,2,3,2,3,1
4	1,10,5,6,3,1	2	1,2,2,2,2,1
...		...	
8	2,3,11,12,9,2	2	3,2,3,2,3,3
...			
12	8,11,17,16,13,7,9,8	2	2,3,3,3,3,3,3,2

Table 7. Switching point counts for zones 8 and 12 as candidates to leave region 2

Zone 8		
Adjacent region table line for Zone 8	2	3,2,3,2,3,3
Switching point count (running total)		0,1,2,3,4,4
Zone 12		
Adjacent region table line for zone 12	2	2,3,3,3,3,3,3,2
Switching point count (running total)		0,1,1,1,1,1,1,2

total). As noted above, zone 8 has four switching points on the current map so cannot be removed from region 2 without loss of contiguity. Zone 12, by contrast, has only two such points so it can be removed.

The general rule is that a zone can be removed from a region without loss of contiguity only if it has two switching points. However, whilst two switching points are necessary they are not sufficient. In addition, there must be a region to which the zone can be transferred other than 'the outside world' (labelled zone 1 and region 1 in the tables). Thus, for example, whilst zone 4 has two switching points, there is no region to which it can be attached.

There is one qualification that should be made to the claim that two switching points are necessary for a zone to be removed from its current region without loss of contiguity. Whenever the set of zones in one region is completely surrounded by the set of zones that constitute another region, it may be possible to remove a zone from the latter region without loss of contiguity. Informally, breaking the ring around the inner zone in one place does not leave the remaining zones in the outer region disconnected. There are two ways of dealing with this issue. The first is to use an initial partition that does not include any regions within regions and to reject the move of any zone that would create a region within a region (by rejecting all moves that would give the transferred zone four switching points). This is frequently acceptable in a political redistricting context because districting legislation or practice often precludes the possibility of accepting regions within regions. In contexts where this rule does not apply, the basic algorithm can be modified to include the setting and checking of two flags – one to indicate regions that ring one or more other regions and another to indicate regions that are ringed. With such flags, the procedure for counting switching points can be modified as follows.

(1) For a target zone in a region that rings (an)other region(s), do not count as a switching point a node which has an internal boundary arc to one side and an arc that is a boundary between the zone's region and (one of the) ringed region(s) on the other.
(2) If a pair of nodes with this property is encountered and the zone is chosen for removal (which would occur only if it had two switching points by this modified counting method) then change the flag on both the outer and inner regions.

As it is possible for a region to be both ringed and ringing, two flag positions are required for each region.

There a number of other complications that can arise, two of which are concerned with topographical features. If a lake is treated as a zone with a population of zero, it is possible to generate a partition that breaks the conventional contiguity rule. For example, if zone 6 is a lake but is treated as a zone and zone 4 is removed, the remaining parts of the light region are not contiguous but the treatment of the lake as a zone makes them appear to be. This problem can be overcome by taking an arbitrary point in the centre of the lake and connecting arcs to it from each of the nodes on the lake edge. By adding each of the newly created sections of the lake to the land zone with which it shares a border, the lake ceases to be a difficulty.

Islands pose a more complex problem. Whenever there are islands, a fully contiguous partition is not possible so it is necessary to abandon the notion of starting with such a partition then maintaining contiguity whilst pursuing optimality. The best that can be done is to make some relatively arbitrary decisions (which, in a political redistricting context, may be guided by legislation or custom and practice) about the mainland zones that constitute the pseudo-neighbours of (the zones on) the island. In terms of the base map, this involves a procedure similar to that used for lakes.

Finally, it is important to note that the switching point method continues to work when the number of arcs meeting at a node is greater than three. Suppose, for example, that the base map is a rectangular grid so that each zone has four neighbours with which it shares an arc boundary and four others with which it shares only a node. Node-only connections are not contiguous so the removal of a zone should be rejected if it would leave its current region in two parts connected only at a node. To achieve this, it is not necessary to modify the method for counting switching points but it is necessary to ensure that the list of adjacent zones includes, in order, those that share an arc border and those that share a node only. This information can be obtained readily in a GIS environment in which an arc-node topology is employed. This argument applies to all nodes with more than three arcs and not just to grid structures.

4 Comparative performance

It is interesting to compare the performance of this new method with the old method of Openshaw and Rao on the small example problem: it required 3,000 DO loops as compared with 350,000; 120,000 assignments as compared with 9.5 million; and 80,000 IF statements as compared with 14 million. These measures are crude and they relate to one problem only but they are indicative of the advantages of the new method.

To test the relative performance of the two methods more systematically, Alvanides and Macmillan [6] ran two test problems. The first used the 930 zone map and partitioned it into a number of regions ranging from 2 to 500. Figure 7 shows the CPU times of the two methods which converge, as one would expect, as the number of regions approaches the number of zones. The second test used a 2310 zone map and a similar range of region numbers (see Fig. 8). Here, the new algorithm showed a significant performance advantage over the old one up to the 500-region level.

5 The Simulated Annealing Redistricting Algorithm (SARA)[1]

Having presented an outline of the algorithm in Sect. 1, the switching-point method in Sect. 3, and some indication of the benefits of that method in Sect. 4, it is now possible to give a fairly succinct, technical statement of the algorithm.

SARA partitions a set of N zones into M regions (where $M < N$) such that each zone is allocated to one and only region, each region has at least one zone, and the zones in each region are contiguous. The algorithm keeps contiguity from an initially contiguous partition and uses a directed, probabilistic, search procedure within a

[1] The intellectual property right in this work is owned by the University of Oxford.

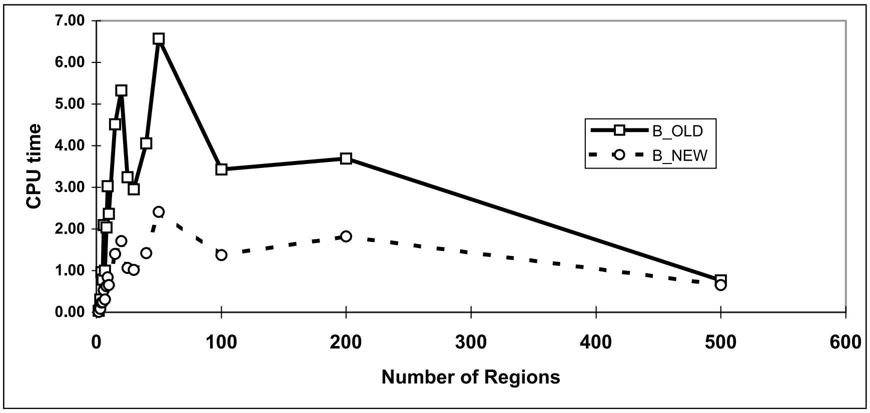

Fig. 7. Comparison of CPU times for the old (Openshaw and Rao) and new contiguity checking procedures for a 930 zone map

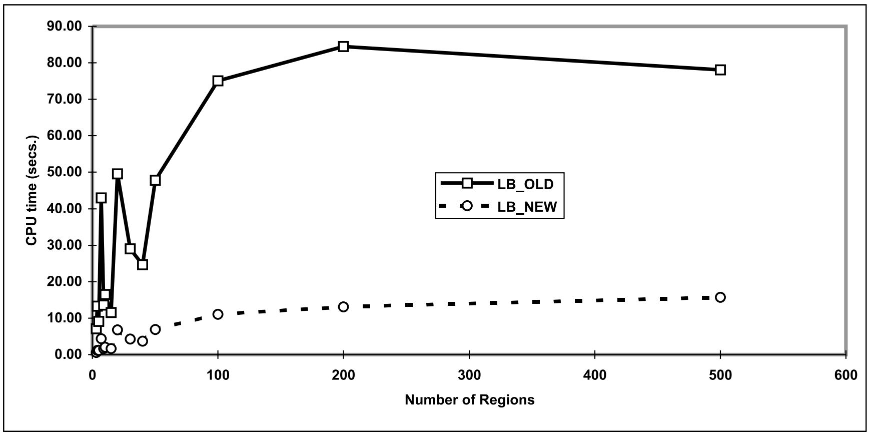

Fig. 8. Comparison of CPU times for the old (Openshaw and Rao) and new contiguity checking procedures for a 2310 zone map

simulated annealing structure to find a partition that minimises the deviation in population between regions.

Data and parameters

$(p_1, \dots, p_N)$ is the vector of populations for zones $n = 1, \dots, N$.

P is the equal-apportionment target population for each region, so $P = \frac{1}{M}\sum_{n=1}^{N} p_n$.

A^Z is the adjacent zone matrix. It has N rows and a number of columns determined by the topology of the zonal boundary map. Column 1 contains the zone identification (*id*) number. Column 2 contains the *id* of an arbitrarily chosen adjacent zone, where *id* number 1 represents the outside world. Column 3 contains the *id* of the next adjacent zone moving around the zone's perimeter in an arbitrarily chosen direction. The next columns contain the *id*s of all other adjacent zones, in order around the perimeter, up to an including the starting zone (the one whose *id* appears in column 2). The remaining columns contain the number −1.

$\alpha_s, \alpha_u, \beta$ and ε are parameters whose values are around 10, 100, 0.95 and 0.05 respectively.

Variables

$(x, \ldots, x_N)'$ is a partition, where $x_n \in \{1, \ldots, M\}$ for $n = 1, \ldots, N$.

$Z_m \equiv \{n | x_n = m\}$ is the set of zones in region m, for $m = 1, \ldots, M$.

A^R is the adjacent region matrix. It has the same form as the adjacent zone matrix. It is derived from that matrix by replacing each adjacent zone *id* by the *id* of the region to which it belongs. Thus, the first column of this matrix is the vector $(x, \ldots, x_N)'$. Similarly, Z_m is the set of row numbers of A^R for which the number in the first column is m.

$(P_1, \ldots, P_M)$ is the vector of regional populations, where $P_m \equiv \sum_{n \in Z_m} p_n$, for $m = 1, \ldots, M$.

$\left(P_1^D, \ldots, P_M^D\right)$ is the vector of regional population deviations, where $P_m^D = P_m - P$, for $m = 1, \ldots, M$.

E is the set of regions with excess populations (positive deviations): $E \equiv \left\{m | P_m^D > 0\right\}$

$P_{m'n}^S$ is the size of the population deviation that would result in region m' from the subtraction of zone n: $P_{m'n}^S \equiv |P_{m'}^D - p_n|$.

$P_{m'}^S$ is the size of the largest such deviation: $P_{m'}^S \equiv \sup_{n \in Z_{m'}} \left\{P_{m'n}^S\right\}$.

$P_{m''n}^A$ is the size of the population deviation that would result in region m'' from the addition of zone n: $P_{m''n}^A \equiv |P_{m''}^D + p_n|$.

P_n^A is the size of the largest such deviation associated with zone n: $P_n^A \equiv \sup_{m'' \in R_n} \left\{P_{m''n}^A\right\}$, where R_n is the set of regions to which zone n could transfer, which is given by the set of numbers in row n of A^R, excluding m'.

T is the annealing 'temperature' variable.

S_s and S_u are, respectively, the number of successful and unsuccessful transfers.

Algorithm

Initialise:

```
Input a feasible partition $(x_1, \ldots, x_N)'$ as the first column in $A^R$.
Compute the other terms in $A^R$.
Compute $(P_1, \ldots, P_M)$.
Compute $(P_1^D, \ldots, P_M^D)$ and identify the members of the set $E$.
Input an initial temperature $T$.
Set $S_s$ and $S_u$ to zero.
```

Step 1: Compute

$$\pi_{m'} = \begin{cases} P_{m'}^D \Big/ \sum\limits_{m \in E} P_m^D & \forall m' \in E \\ 0 & \forall m' \notin E \end{cases}$$

Select a donor region m' with probability $\pi_{m'}$.

Step 2: For the selected region, m', compute

$$P^S_{m'n} = |P^D_{m'} - p_n| \quad \forall n \in Z_{m'}$$
$$P^S_{m'} = \sup_{n \in Z_{m'}} \{P^S_{m'n}\}$$
$$\pi_{n'} = \frac{P^S_{m'} - P^S_{m'n}}{\sum_{n \in Z_{m'}} \left(P^S_{m'} - P^S_{m'n}\right)} \quad \forall n' \in Z_m$$

Select a zone n' for removal from m' with probability $\pi_{n'}$.

Step 3: For the selected zone n', use row n' of A^R to check for loss of contiguity if n' is removed from m'. Reject move if contiguity would be lost and return to Step 2.

Step 4: Using row n' of A^R, identify the set of regions, $R_{n'}$, to which zone n' could transfer ($R_{n'}$ is the set of numbers in row n' of A^R, excluding m'). Compute:

$$P^A_{mn'} = |P^D_m + p_{n'}| \quad \forall m \in R_{n'}$$
$$P^A_{n'} = \sup_{m \in R_{n'}} \{P^A_{mn'}\}$$
$$\pi_{m''} = \frac{P^A_{n'} - P^A_{m''n'}}{\sum_{m \in R_{n'}} \left(P^A_{n'} - P^A_{mn'}\right)}$$

Select a recipient region m'' for the selected zone n' with probability $\pi_{m''}$[2].

Step 5: (a) If

$$P^S_{m'n'} + P^A_{m''n'} < P^D_{m'} + |P^D_{m''}|$$

then accept the transfer[3] and let $S_s = S_s + 1$.

(b) If the above condition is not satisfied, then accept the transfer with probability

$$\pi = \exp\left[-\left(P^S_{m'n'} + P^A_{m''n'} - P^D_{m'} - |P^D_{m''}|\right)/T\right]$$

Step 6: If the transfer is accepted, recompute $\left(P^D_1, \ldots, P^D_M\right)$, adjust A^R, and let $S_s = S_s + 1$. A^R is adjusted as follows: read along row n' of A^Z starting at column 2; go to the row indicated by the number in this column; move along this row until the number n' is found; alter the value in the corresponding cell in A^R from m' to m''; go back to row n' of A^Z, move to the next column and repeat until the number that appears in column 2 is encountered again. Alter the number in column 1 of row n' of A^R from m' to m''.

If the transfer is not accepted, let $S_u = S_u + 1$.

[2] Note that if $P^D_{m''} > 0$ for all m'', a region will still be chosen. An alternative version of this step is to assign equal probabilities to the alternative destination regions in $R_{n'}$.

[3] Note that $P^D_{m'} > 0$ since m' is selected from E.

Step 7: If $\sum_m |P_m^D| < \varepsilon MP$ then stop.

If the number $S_s < \alpha_s N$ and $S_u < \alpha_u N$ then return to Step 1.
If not, then let $T = \beta T$ and return to Step 1.

6 Variations and extensions

It is possible to produce a number of variations and extensions to the algorithm, one of which (involving flags for regions that surround or are surrounded by other regions) has been noted already. Another variation involves the use of a different type of objective function. For example, it would not be difficult to adapt the algorithm to tackle problems of spatially constrained classification, where the objective is to minimize the ratio of within-region to between-region variations[4]. One possible extension is to introduce additional design criteria. The extra criteria might relate to spatial properties of regions (such as compactness), to landscape features (which might be represented by different types of zone boundary sections), or to non-spatial properties of zones and regions (in addition to population size). Such criteria could be dealt with either as additions to the constraints under which the system of zone transfers operates or as supplementary objectives. In the latter case, the various objectives could be weighted *ex ante* to produce a single operational objective. Alternatively, a set of partitions could be generated using a range of arbitrarily chosen weights from which a set of efficient partitions could be identified. For each such partition, there would be a range of *ex post* weights under which the partition would be optimal (effective as well as efficient), allowing a judgement to be made about the acceptability of weight ranges and, by implication, the best compromise solution.

Whatever sophistications are introduced at this level, the whole procedure remains dependent upon the efficiency of the contiguity checking routine.

References

1. Openshaw S, Rao L (1995) Algorithms for reengineering 1991 Census geography. *Environment and Planning A* 27: 425–446
2. Pierce T (1992) *A GIS-compatible, active computer algorithm for American Congressional redistricting*. D.Phil. thesis, School of Geography, University of Oxford
3. Macmillan W, Pierce T (1994) Optimization modelling in a GIS framework: the problem of political redistricting. In: Fotheringham S, Rogerson P (eds) *Spatial analysis and GIS*. Taylor and Francis Ltd., London, pp 221–246
4. Bowdry M (1990) Simulated annealing – an improved computer model for political redistricting. *Yale Law and Policy Review* 8: 163–179
5. Macmillan W, Pierce T (1996) Active computer-assisted redistricting. In: McLean I, Butler D (eds) *Fixing the boundaries*. Dartmouth Publishing Company, Brookfield Vermont, pp 219–234
6. Alvanides S, Macmillan W (1997) A fast contiguity checking algorithm for use in zone design (unpublished conference paper) *GISRUK '97*, Leeds

[4] The author is grateful to one of the referees for making this and other helpful points.