A DISTRICTING EXPERIMENT WITH A CLUSTERING ALGORITHM

Lawrence D. Bodin

Urban and Policy Sciences Program
State University of New York
Stony Brook, New York 11790

INTRODUCTION

In this paper, we consider the problem of how to partition a geographic area into a collection of contiguous, approximately equal population districts. This problem is analogous to the graph theory problem of partitioning a network into a fixed number of contiguous sections such that the sum of the node weights within each sector is the same. In experimenting with this heuristic algorithm, we found that the procedure is extremely fast computationally, does a reasonable job of partitioning the network, but does not terminate at what could be called a strong set of local optima. (The question of when local optima become global optima, as well as the various properties of heuristic algorithms that this question implies, is being considered by S. Savage in his Ph.D. thesis at Yale University.)

Thus, the algorithm terminates at one out of many solutions that do approximately the same job. The heuristics that make up this algorithm, however, are not discriminating enough to pick out the best of these solutions. Nevertheless, we derive some very useful benefits from this procedure. First, the procedure quickly makes a good initial guess regarding an optimal partition. This guess can then be used as an upper bound to make exact, but more time-consuming, algorithms more efficient. Second, since the algorithm generates many good, approximately equal partitions cheaply in terms of computer time, each of these solutions can then be examined for other attributes (such as compactness) and the best of these solutions accepted.

Other attempts to do redistricting and graph partitioning have been made in the past. There is also a close relationship between the redistricting problem and a class of facility location problems. Some of the studies in these areas are Hess and Weaver,[1] Garfinkel,[2] Marazana,[3] Trehan,[4] Teitz and Bart,[5] Marks and colleagues,[6] Spielberg,[7, 8] Scott,[9] and Revelle and Swain.[10]

THE ALGORITHM

We are given a geographic region broken down into subregions such as counties or precincts. We now interpret this geographic region as a directed network, as follows:

1. Each subregion in the area becomes a node in the network.
2. Two nodes, k and l, are connected in the network if their corresponding subregions have a boundary in common. We draw two branches connecting nodes k and l, one oriented out of k and the other oriented out of l.
3. The weight on the branch from node k to node l, d_{kl}, is the population of subregion l; the weight on the branch from node l to node k, d_{lk}, is the population of subregion k.*

* A second way to take care of this rule for constructing the network is to assume that there is a weight for each node equal to the population of the corresponding subdistrict. I feel, however, that the explanation in the text is a clearer description of the algorithm and embodies more of the spirit of the labelling algorithms in network flow theory.

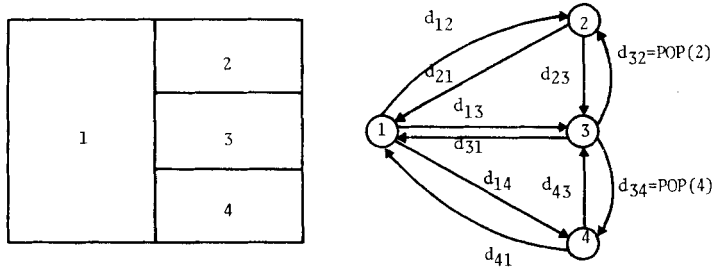

FIGURE 1.

The construction of the network using these rules is illustrated in FIGURE 1.

The algorithm is divided into two steps. The first step, which is in the spirit of the classical node labelling algorithms in network flow theory (see, for example Ford and Fulkerson[11]), results in a first guess as to a reasonable partition. The second step, which can be thought of as an adaptation of the branch exchange technique of Lin,[12] improves the partitioning derived in the first step. Once the nodes are labelled, the idea behind this algorithm is to reduce the difference in population between two partitions, with the objective of converging to a set of partitions that are approximately equal in population.

Step 1

To begin step 1, we assume that we know N, the number of partitions into which we are to divide the geographic area. In political redistricting this is a realistic assumption. We then assume node n_j to be the center of the jth partition, $j = 1, 2, \cdots, N$. We also assume that in choosing our centers, $n_j \neq n_k, j \neq k, j = 1, 2, \cdots, N, k = 1, 2, \cdots, N$.

For each node, m, in the network, we assign a three-dimensional label, $l(r_m, s_m, t_m)$ where r_m is the cluster number to which node m is assigned, $1 \leq r_m \leq N$; s_m is the weight of cluster r_m, $1 \leq r_m \leq N$; t_m is the predecessor node to node m in the tree that is constructed out of the center node in cluster r_m. Initially, $r = 0$, $s = \infty$, and $t = 0$ except for all nodes n_j (the center nodes), where $r = j$, $t = -1$, and s is the population of the subdistrict associated with node n_j.

We then search through all the nodes in the network. Step 1 of the algorithm terminates when a pass through all the nodes in the network does not result in changing the label on any node. We shall now explain the conditions under which the label on a node can be changed.

We say a node is labelled if its first subindex $r \neq 0$ and is unlabelled if $r = 0$. Initially, only the center nodes are labelled. Now, consider a node m that is labelled, and let n be a node adjacent to node m in the network. One of two conditions can arise: either node n is unlabelled or node n is labelled.

Condition 1

When node n is unlabelled, we set the labels for node n as follows:

$$r_n = r_m$$
$$t_n = m$$
$$s_n = s_m + d_{mn}$$

where d_{mn}, the population of subdistrict n, is the distance from node m to node n. Furthermore, we change the s label on all nodes whose r label is the same as r_m to

be equal to s_n. Thus, the s label indicates the total population assigned to the partition denoted by r_m.

Condition 2

When node n is labelled and $r_n \neq r_m$ (if $r_n = r_m$, then nodes n and m are in the same partition, and no further analysis of this pair of nodes is necessary). In this case we must determine if by placing node n in the partition denoted by r_m, we break the r_n partition into two pieces. To do this, we backtrack from every node p in the cluster denoted by r_n to the original center of the cluster, using the t label on the nodes. If we arrive at the origin without encountering node n, then we conclude that the particular node p that we are examining is not a successor to node n. If, in backtracking from node p, we encounter node n, then we let $z = z + d_{t_p p}$, where $d_{t_p p}$ is the distance from node t_p to node p. z keeps track of the total population of that part of the tree that is made up of node n and its successors. The key to the above is that we have generated a tree out of every center. Hence, every node has a unique predecessor, which is indicated by the t sublabel.

Before we make any exchanges between partitions, we check the relation between $z + s_m$ and s_n. If $z + s_m > s_n$, then we do not make any exchanges, since the populations in these two partitions would be further spread apart than is indicated by the present situation. If $z + s_m < s_n$, then an exchange is carried out. To make this exchange, we relabel the nodes as follows:

$t_n = m$.

For all nodes q whose r label is r_n, set $s_q = s_q - z$.

For node n and all nodes that are successors to node n in the partition denoted by r_n, set the r label to r_m.

For all nodes q whose r label is r_m, set $s_q = s_q + z$.

This relabelling exchanges node n and all of its successor nodes from its original partition to the partition denoted by r_m. FIGURE 2 is a pictorial representation of this situation.

Step 2

Step 2 is an adaptation of the branch exchange procedure of Lin.[12] Let node n be in the l_1 partition and node m be in the l_2 partition. Step 2 of this procedure attempts to make one of the following three exchanges (see FIGURE 3):

Exchange 1: Place node n in the l_2 partition.

Exchange 2: Place node m in the l_1 partition.

Exchange 3: Place node n in the l_2 partition and node m in the l_1 partition.

To carry out any of the above three exchanges, one must insure that the partitions formed after the exchanges are made are contiguous. Let d_0 be the absolute value of the difference in population between l_1 and l_2 before any exchanges are made. Let d_i be the absolute value of the difference in population after carrying out exchange i ($i = 1, 2, 3$) where $d_i = \infty$ if the exchange results in a region which is not contiguous. Then carry out exchange k if

$$d_k = \text{Min}(d_1, d_2, d_3) < d_0.$$

In this transformation we insure that both of the above partitions remain contiguous and that the difference in population between these two regions is reduced. We note in Step 2 that we have disregarded the tree structure generated in Step 1, and we have insured that the contiguity of the partitions are maintained. As with Step 1, we repeatedly search through all the nodes in the network. We terminate Step 2 when a pass through all the nodes in the network gives no exchanges.

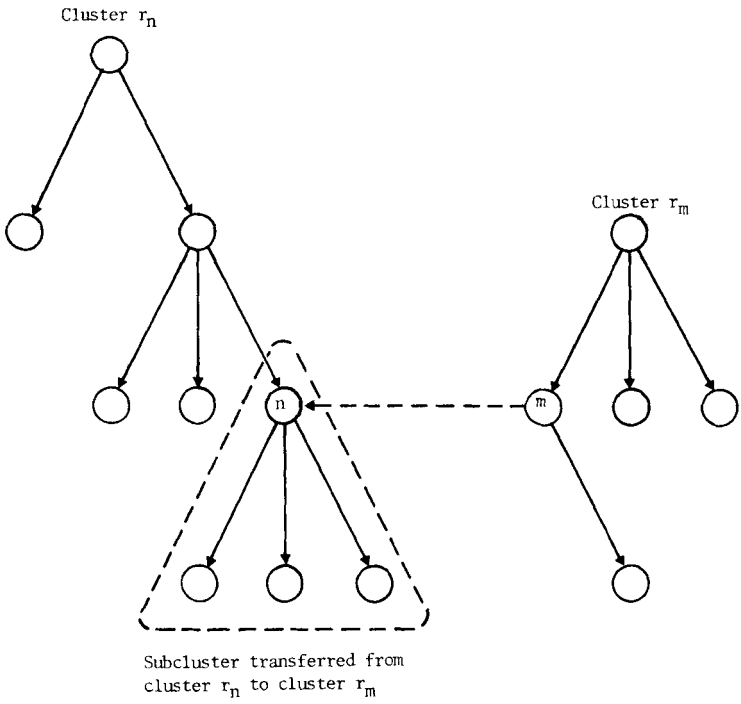

FIGURE 2.

This algorithm can be extended to include more extended branch exchange procedures such as in Lin[12] or Lin and Kernighan.[13] These extended procedures were not implemented in our experimental code. The results given below were derived using a code based on the algorithm described in Steps 1 and 2 above.

COMPUTER RESULTS

We tested the algorithm on the 1960 census data for the state of Arkansas. Arkansas is made up of 75 counties whose populations range from 4,927 in Perry County to 242,980 in Pulaski County.

We ran the code on the university's IBM 370 Model 155 computer, assuming that the region was to be divided into three districts, five districts, and nine districts. For three districts and five districts, we ran the program for five minutes CPU time; for nine districts, we ran it for ten minutes CPU time. We used a random number generator to assign the centers, except that Pulaski County, because of its population, was assumed to be a center. For three centers we found 122 trial solutions, for five centers we computed 75 trial solutions, and for nine centers we found 82 trial solutions. Thus, as can be seen, this code is exceptionally efficient.

We decided that a reasonable function that measures the dispersion of a particular solution would be the difference between the largest and smallest subdistrict formed in the final partition, divided by the expected population of the subdistrict. This function measures the percentage of difference between the largest and smallest partition found on a particular trial. We then computed this percentage measure of

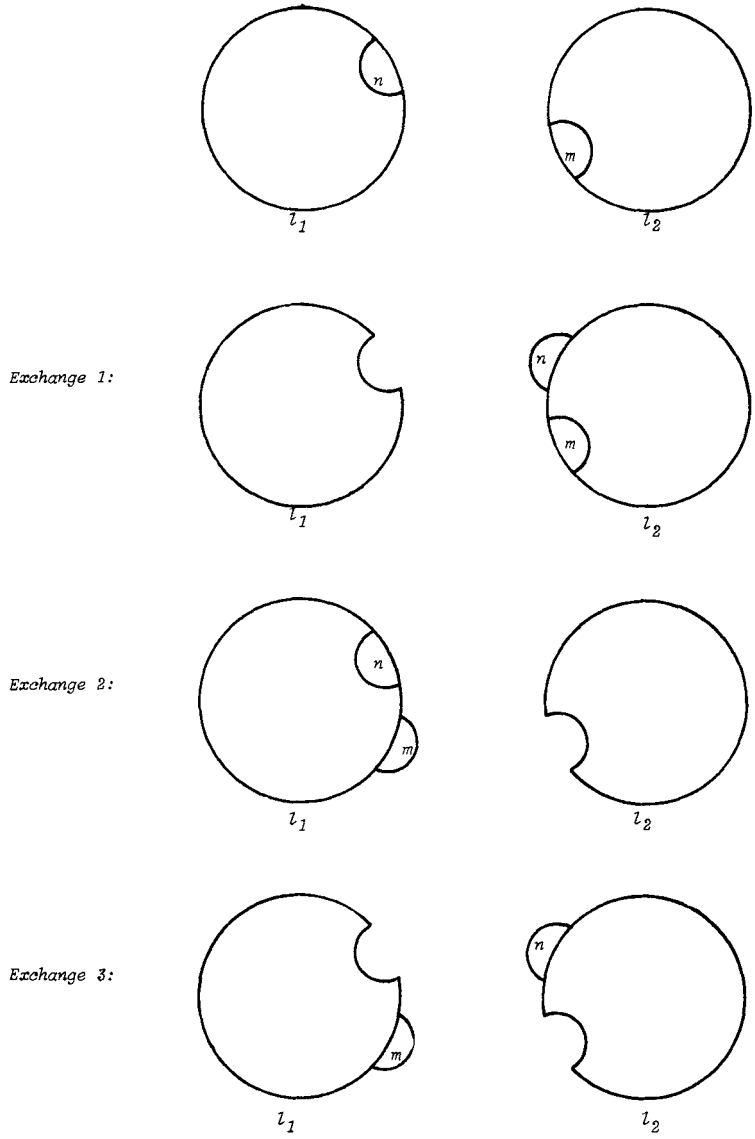

FIGURE 3.

dispersion for each of the trial solutions and developed a histogram for the measure of dispersion based on these trial solutions for the three, five, seven, and nine districts. These histograms are given in TABLE 1.

It is evident from these results that the algorithm works best when the number of subdistricts into which the district is to be divided is small compared to the number of subdistricts initially given. However, even in the case where there are nine sub-

TABLE 1
HISTOGRAMS FOR PERCENTAGE MEASURES OF DISPERSION IN THREE, FIVE, AND NINE DISTRICTS

Percentage Spread in Clusters	3 Centers	5 Centers	9 Centers
0–.5	113	10	0
.5–1.0	6	34	0
1.0–1.5	2	16	1
1.5–2.0	1	9	1
2.0–2.5	0	2	5
2.5–3.0	0	1	10
3.0–3.5	0	0	10
3.5–4.0	0	1	10
4.0–4.5	0	1	4
4.5–5.0	0	0	3
5.0–5.5	0	0	5
5.5–6.0	0	1	9
6.0–6.5	0	0	3
6.5–7.0	0	0	3
7.0–8.0	0	0	6
8.0–9.0	0	0	5
9.0–10.0	0	0	1
>10.0	0	0	6

districts to be found, 75 initial subdistricts, and a skewed distribution of population in each of the initial subdistricts, many of the trial solutions found using the algorithm developed in this paper were quite acceptable.

REFERENCES

1. HESS, S. W., J. B. WEAVER, H. J. SIEGFELDT, J. N. WHELAN & P. A. ZITLAU. 1965. Nonpartisan political redistricting by computer. Operations Res. **13**(6): 998–1006.
2. GARFINKEL, R. 1970. The set covering problem. Working Paper Ser. S 7024. Systems Analysis Prog. Univ. Rochester. Rochester, N. Y.
3. MARAZANA, F. E. 1964. On the location of supply points to minimize transport cost. Operational Res. Q. **15:** 261–270.
4. TREHAN, S. 1971. The facility location problem—a survey. M.A. Thesis. State Univ. New York. Stony Brook, N. Y.
5. TEITZ, M. B. & P. BART. 1968. Heuristic methods for estimating generalized vertex median of a weighted graph. Operations Res. **16:** 955–961.
6. MARKS, D. H., C. REVELLE & J. C. LIEBMAN. 1970. An analysis of private and public sector location models. Management Sci. **16:**
7. SPIELBERG, K. 1969. An algorithm for the simple plant-location problem with some side conditions. Operations Res. **17:** 85–111.
8. SPIELBERG, K. 1968. Plant location with generalized search origin. IBM Technical Rept. No. 320-2929.
9. SCOTT, A. J. 1970. Location-allocation systems: a review. Geograph. Anal. **2**(1): 95–118.
10. REVELLE, C. S. & R. W. SWAIN. 1970. Central facilities location. Geograph. Anal. **2**(1): 30–42.
11. FORD, L. B., JR. & D. FULKERSON. 1962. Flows in Networks. Princeton Univ. Press. Princeton, N. J.
12. LIN, S. 1965. Computer solutions of the travelling salesman problem. Bell System Tech. J. **44:** 2245–2269.
13. LIN, S. & B. W. KERNIGHAN. 1972. A heuristic algorithm for the travelling salesman problem. Computer Sci. Tech. Rept. No. 1. Bell Telephone Labs. Murray Hill, N. J.