

To the Graduate Council:

I am submitting herewith a thesis written by L. Scott Buckels entitled "Optimal Political Districting: An Interactive Computer Graphics System." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Management Science.

Kenneth C. Gilbert

Kenneth Gilbert, Major Professor

We have read this thesis and recommend its acceptance:

Bruce A. Ralston
Dale R. Fox

Accepted for the Council:

Lawrence Munkel

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under the rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that the accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature L. Jean Baker

Date 14 DECEMBER 1989

**OPTIMAL POLITICAL DISTRICTING:
AN INTERACTIVE COMPUTER GRAPHICS SYSTEM**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

L. Scott Buckels

May 1990

ABSTRACT

An interactive computer graphics system is developed for application to the solution of political districting problems. The foundation for the system is a previously developed two phase algorithm which finds all optimal solutions, for a given set of criteria, to the districting problem. The first phase generates all feasible districts while the second phase selects the optimal plan. The optimal solution is the set of districts covering the entire area and minimizing the maximum deviation of any district population from the mean district population.

The interactive graphics system allows input of map points for any area, determination of the optimal districting plan for an area, and graphic display of the map and optimal solution. Additionally, during execution of the graphics routine, specific boundaries may be interactively selected as either necessary or inappropriate to exist in a solution. The system determines the new optimal plan based on the additional criteria specified during the interactive graphics routine, and displays the new map and optimal solution.

Computational and graphical results are given for real problems. Additionally, enhancements to the system are considered.

Table of Contents

Chapter 1 BACKGROUND	1
1.1 INTRODUCTION	1
1.2 HISTORY	2
1.3 CRITERIA	3
1.4 LITERATURE SURVEY	5
1.5 COMPUTER GRAPHICS	7
 Chapter 2 THE ALGORITHM	 9
2.1 INTRODUCTION	9
2.2 BOUNDARIES	10
2.3 THE PHASE I MODEL	12
2.4 THE PHASE II MODEL	16
2.5 THE PHASE I ALGORITHM	17
2.6 THE PHASE II ALGORITHM	20
2.7 CONCLUSION	22
 Chapter 3 THE SYSTEM	 24
3.1 INTRODUCTION	24
3.2 GENERAL FLOW OF THE IGPDS	26
3.3 PROGRAM DESCRIPTIONS	29
3.4 DATA FILES	34
3.5 USE OF THE SYSTEM	40

Chapter 4 RESULTS	47
4.1 INTRODUCTION	47
4.2 NEVADA	47
4.3 NEW MEXICO	55
4.4 WEST VIRGINIA	61
4.5 WASHINGTON	71
4.6 CONCLUSIONS	81
 LIST OF REFERENCES	 83
 VITA	 86

List of Figures

Figure 2-1: Example of two types of boundaries.	12
Figure 3-1: Schematic Flow Diagram for the Interactive Graphics Political Districting System (IGPDS).	27
Figure 3-2: Opening menu for the IGPDS.	42
Figure 3-3: Data file creation menu for the IGPDS.	43
Figure 3-4: Sample graphics display and menu.	45
Figure 4-1: Nevada current plan.	49
Figure 4-2: Nevada - Run 2 results.	51
Figure 4-3: Nevada - Run 3 results.	52
Figure 4-4: Nevada - Run 4 results.	54
Figure 4-5: New Mexico current plan.	57
Figure 4-6: New Mexico - Run 1 results.	61
Figure 4-7: New Mexico - Run 2 results.	61
Figure 4-8: New Mexico - Run 3 results.	62
Figure 4-9: West Virginia current plan.	65
Figure 4-10: West Virginia - Run 1 results.	67
Figure 4-11: West Virginia - Run 2 results.	68
Figure 4-12: West Virginia - Run 3 results.	69
Figure 4-13: Washington current plan.	73
Figure 4-14: Washington - Run 1 results.	76
Figure 4-15: Washington - Run 2 results.	77
Figure 4-16: Washington - Run 3 results.	79

Chapter 1

BACKGROUND

1.1 INTRODUCTION

Earl Warren stated that "The most important Supreme Court rulings of [his] sixteen years as Chief Justice of the United States were those declaring that one man's vote should mean as much as any other man's." (Grofman, 1982, p. xiii) The Supreme Court reached several landmark decisions concerning the equitable legislative representation of the general population, creating the need for impartial and consistent methods for determining political districts.

An interactive graphics system for solution of the political districting problem is described in the remainder of this paper. The basis for the development of this system is the work done on the redistricting problem by Garfinkel (Garfinkel, 1968). The original work by Garfinkel included algorithmic development and implementation of the algorithm into assembly language for the IBM 7094.

The development of the new system attempted to follow three main objectives. First, the new system was implemented using only higher level languages to increase the transportability of the system and make the system more accessible. Secondly, the inclusion of menus and interactive graphics routines meets the objective of making the system user-friendly and the results easily understandable. Finally, and perhaps most importantly, was the objective to develop a system that would solve large problems. The original work by Garfinkel could not solve large problems, due to the size and speed of the computer and the

demonstrated exponential growth of the first phase of the algorithm. Implementation of the new system on a faster and more powerful computer attempts to overcome some of this size limitation. Allowing the user to specify boundaries that must be included in the final solution attempts to overcome the size limitation by curbing the exponential growth of the algorithm by breaking large problems into smaller ones. Since boundary specifications alter the original problem, this solution process may result in a solution that is sub-optimal to the original problem. A discussion of the use of boundaries to limit problem size is given in Chapter 2.

The next section of this chapter briefly summarizes some of the history of legislative districting and the related court cases. Next is a short description of the criteria used for determining an equitable districting plan. The following section reviews some other approaches in the literature to the political districting problem. The final section describes the advantages that a computer graphics system provides to the districting process.

1.2 HISTORY

In a 1962 Tennessee case, *Baker v. Carr* (1962), the Supreme Court established the general principle of population equality among districts. In a 1964 Georgia case, *Wesberry v. Sanders* (1964), the equality principle was applied to congressional districts, and the phrase "one-man, one-vote" was introduced. An Alabama case, *Reynolds v. Sims* (1964), helped define equal population when the court ruled that both houses of a state legislature must be apportioned on the basis of "substantially equal" population.

The idea of population equality among legislative districts is not a new one in the United States. As early as 1787, the Northwest Ordinance, providing for the future governments of the territory west of the Appalachians, stipulated that its

inhabitants "shall always be entitled to the benefits . . . of a proportionate representation of the people in the legislature." (Goldwin, 1968, p. 72)

Throughout the nineteenth century, as new states were formed and the constitutions of older ones were revised, apportionment on a population standard became widespread. Even in those relatively few states where population was completely ignored as a basis for representation, the early disparities were often quite small. (Goldwin, 1968, pp. 72-73)

Toward the end of the nineteenth century a distrust of growing cities resulted in restrictions on urban power in many states, often in the form of constitutional modifications. (Goldwin, 1968, p. 73) In about one third of the states, all counties were entitled to one senator and at least one representative, irrespective of population. In a majority of states, constitutional provisions to redistrict were simply ignored for generations. (Morrill, 1981, p. 9)

The extreme inequity of representation, the prevalence of gerrymandering, and a more activist Supreme Court led to the rulings in the historic reapportionment cases in the 1960's. These rulings resulted in the need for redistricting in most states of the Union. It also established the need for a systematic method of determining districts based on stated criteria. (Garfinkel, 1968, p. 3)

1.3 CRITERIA

The political districting process divides an area, such as a state, into smaller areas (districts), each of which is assigned a number of representatives. Traditionally, the state has been partitioned into a number of indivisible population units, such as counties, census tracts, or enumeration districts, each of which must be assigned to one and only one district. The resulting districting scheme is called a plan.

The determination of an equitable districting plan must be based on some set of standardized criteria. Following is a discussion of criteria that are generally accepted as necessary for developing a fair districting plan.

Population

The principle of one man, one vote implies an ideal of legislative districts with equal population, although absolute equality is neither possible nor expected. (Baker, 1973, p. 202) Although the constitution has not changed, the enforcement or recognition of equality of population has only occurred since the historic court decisions. (Morrill, 1981, p. 18) It is generally accepted that an acceptable and equitable districting plan implies minimal population deviation between districts in the plan. In fact, the second phase of Garfinkel's algorithm minimizes the maximum deviation of any district population from the mean district population.

Contiguity

A single criterion of population equality for determining a districting plan can easily result in a plan that radically violates the spirit and rationale of one man, one vote. The traditional descriptive constraints have included the typical requirements that districts be composed of compact and contiguous territory. (Baker, 1973, p. 204) Garfinkel loosely defines a contiguous district as one in which it is "possible to walk from any point in the district to any other point in the district without crossing another district". (Garfinkel, 1968, p. 6)

Compactness

As a result of critics and reformers having stressed compactness both as inherently desirable and as a defense against gerrymandering, it has become a major criterion for evaluating districting plans. (Morrill, 1981, p. 21) While definitions and measurements of compactness vary, "a district is thought to be geographically compact if it is somewhat circular or square in shape rather than

long and thin". (Garfinkel, 1968, p.6) Morrill argues that compactness measures other than geometric shape be utilized, including the ratio of maximum to minimum travel time around the district. (Morrill, 1981, p.22)

Other Criteria

In addition to the criteria of population equality, contiguity, and compactness several other criteria have been suggested. These include maintaining county boundaries as much as possible, preserving the integrity of communities of interest, minimizing the number of incumbents who will lose their seats, and guaranteeing each party a core of safe seats. For a more thorough discussion of the criteria involved in districting plans, see Morrill (1981) or Baker (1973).

1.4 LITERATURE SURVEY

Because of the very large amounts of demographic and political data, and because of the desire to measure and compare alternative plans, computers have been employed in the redistricting process. While several algorithms in the literature have been implemented on computer systems for solution of the districting problem, little attention has been focused on a graphical representation of the solution.

Most of the districting methods described in the literature are heuristic, one exception being the algorithm described in this paper (See Garfinkel, 1968, and Garfinkel & Nemhauser, 1970). These heuristic approaches have the advantage of speed resulting from limited, straightforward computations, but provide no guarantee of reaching an optimal solution. But because of their computational efficiency, such methods are suitable for problems with many population units. Some of these approaches to the districting problem are well documented and referenced. (See Chance, 1965, Forrest, 1964, Kaiser, 1964, Nagel, 1965).

One approach (Hess et al., 1965) uses a 'warehouse-location' heuristic, requiring an initial guess for district centers and the use of a transportation algorithm, to map compact and contiguous districts of equal population. The population moment of inertia, the summed squared distances from each person to his district's center, is the minimization criterion and compactness measure.

Substantial work in the area of legislative districting, in particular for the State of Iowa, has been done by Liittschwager. (See Thoreson and Liittschwager, 1967, Gearhart and Liittschwager, 1969, and Liittschwager, 1973). The original work by Thoreson and Liittschwager (1967) uses an arbitrary corner population unit and adds on contiguous units, nearest to the starting unit, until a district meeting the population requirements is created. The follow up work by Gearhart and Liittschwager (1969) implemented the algorithm, with improvements, on a more powerful computer. The algorithmic improvements include a new termination rule for stopping the addition of population units and the ability to easily maintain homogeneity within a county or conform to unusual natural or political boundaries. The final improvement more efficiently checked for enclaves, which are isolated groups of units with too little population to form a district.

Bodin (1973) represents the area to be districted as a network, with the population units serving as the nodes and the arcs representing shared boundaries between the units. A two step algorithm is used to partition the network into a fixed number of contiguous sections. Resulting in a first guess at a partition, the first step of the algorithm consists essentially of a node labelling algorithm. Improvement of this partition is achieved in the second step, by reducing the difference in population between two partitions, converging to a set of partitions with approximately equal populations.

Defining blockfaces as one side of a city block and using them as population units, Savas (1971) develops an algorithm for forming efficient election districts.

Blockfaces are prioritized based on their location relative to the reference blockface and the first step of the algorithm adds additional blockfaces (population units) to the reference unit, which is the first unit selected for that district, based on this priority. The second step of the algorithm checks the new district population and, as necessary, adds more blockfaces, via the first step, or saves the district either with the complete new blockface or a split portion of the blockface.

Tobler (1973) describes a continuous transformation for use in solution of the districting problem. The transformation is modeled as a system of differential equations, but is actually implemented as an iterative finite difference method. This transformation is applied to a map of the area to be districted and the resulting cartogram is overlaid with a square grid, containing a square for each district in the plan. Next, the boundaries of the counties (or population units) are matched as closely as possible, in the least squares sense, to the squares in the grid. Finally, the lines are transferred back to the original map. While this approach does incorporate a graphic representation of the area under consideration, results from this work are difficult to achieve and interpret and may or may not meet all of the generally accepted criteria for a political district.

1.5 COMPUTER GRAPHICS

Because of the large amounts of demographic and political data, and because of the desire to measure and compare alternative plans, computer utilization in the solution of the districting process is desirable. Increased efficiency of data storage and retrieval and shortened turnaround time to develop a districting plan are advantages that computer systems provide to the districting process. The combination of these advantages with the faster perception, more vivid display of data relationships, and simplification of complex data associated with a computer

graphics system creates an effective, systematic method for determining the best districting plan.

Graphics have been used for years to effectively communicate information and aid comprehension of complex information. Because patterns and shapes are inherently less abstract than numbers and languages, "pictures convey information more readily and permit easier retention than textual, verbal, or numerical representations of the same information." (House, 1982, p. 27) Studies show that images can be processed more rapidly than tabular data, "that . . . visual senses can handle tremendous data rates, as high as 72 million words per minute" compared with the 200 to 350 words per minute that most humans read. (House, 1982, p. 15)

Since the fundamental nature of the political districting process includes large amounts of data as well as maps of areas to be districted, including cities, counties, and states, the advantages of developing an interactive computer graphics system for use in the districting process are obvious.

The remainder of this paper describes such an interactive computer graphics system. The next chapter describes the details of the algorithm upon which the system is based. The following chapter describes the interactive graphical political districting system (IGPDS), its general flow, its implementation, and its programs and data files. The final chapter demonstrates some results generated by the system, conclusions reached, and potential areas for further work.

Chapter 2

THE ALGORITHM

2.1 INTRODUCTION

As previously mentioned, the foundation for the development of the software for graphical representation and interaction in solving the political districting problem is the algorithm developed by Garfinkel (1968). Following is a discussion of the model and algorithm used for development of the current system. For a complete, detailed description of the original work, see Garfinkel (1968).

The two phase algorithm finds all optimal solutions, for a given set of criteria, to political redistricting problems. In the first phase, all feasible districts are generated using indivisible population units, such as counties or census tracts. Feasibility in the first phase requires meeting indicated standards for compactness, contiguity, population deviation, and population unit grouping. Certain groupings of population units may be required in the final solution as the result of the interactive graphics routine. These requirements are discussed in detail below.

The second phase of the algorithm searches the feasible districts generated by the first phase to find the set of districts covering each population unit exactly once, while minimizing the maximum deviation of any district population from the mean district population. There are N population units and the number of feasible districts is S . Each district is assigned one representative and there are to be M districts.

The existence of boundaries, the types of boundaries, and the use of boundaries in the solution of the political districting problem are discussed in the

ensuing section. The remainder of the chapter contains a detailed description of the model and both phases of the algorithm.

2.2 BOUNDARIES

Boundaries are implicit from the definition of any district, whether that district is a political district, sales district, or school district. The definition of district boundaries in shaping political districts can have major impact in future elections and therefore in governmental policy. The feasibility criteria of population equality, contiguity, and compactness tend to reduce the opportunity for gerrymandering and increase the "fairness" of the ultimate districting plan. However, during district generation these constraints do not require district boundaries to follow any pre-existing boundaries that may be considered desirable. Such boundaries may be natural boundaries, such as a mountain range or large lake or very strong, traditional ethnic boundaries. As a result, in addition to the constraints for population equality, contiguity, and compactness, the algorithm for the system developed includes a district boundary (or population unit grouping) constraint.

While a district boundary constraint allows the algorithm to recognize and maintain certain desirable boundaries, the most important reason for including such a constraint is as a computational aid. The first phase of Garfinkel's algorithm, as is typical with enumeration algorithms, tends to grow exponentially as the number of population units increases. By specifying certain district boundaries prior to entering the first phase of the algorithm, the possible combinations of population units in a districting plan is limited. These boundary specifications limit the potential growth of the problem by effectively reducing larger problems into smaller ones. In other words, without the boundary specifications, the algorithm enumerates *all* feasible districts with regard to

population equality, contiguity, and compactness. The boundary specifications add the additional constraint of certain population unit groupings and, as a result of this constraint, there are fewer feasible districts for the algorithm to find and the solution process becomes more manageable.

It is important to note that this increased manageability in the solution process is not without cost. Given enough time and memory on a computer, Garfinkel's original algorithm finds the optimal districting plan for any given problem. With the addition of any district boundary specifications, the nature of the original problem is altered. While the current algorithm finds the best plan given the additional boundary constraint(s), the solution found may be sub-optimal to the original problem.

With respect to the districting system described below, two types of boundaries exist. First, a boundary between two population units may prohibit those population units from existing in the same district. Secondly, a boundary may exist between two population units that does not necessarily prohibit the two units from co-existing in a district. For example, for the boundary highlighted in Figure 2-1, the first type of boundary prohibits units 1 and 33 from existing in the same district. The second boundary type allows a district to be built that goes around that boundary and includes both units. The system discussed herein allow only the first type of boundary. That is, if a boundary is specified as existing as a district boundary, the two units that share that boundary *cannot* exist in the same district. As a further computational aid, the system allows boundaries to be selected that cannot exist as a district boundary. In other words, the two units that share that boundary *must* exist in the same district.

The constraints relating to boundary specifications are added to Garfinkel's original algorithm in a practical attempt to solve large problems. With adequate computer resources, the original algorithm is guaranteed to find the optimal

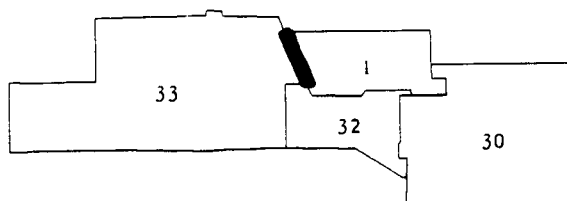


Figure 2-1: Example of two types of boundaries.

districting plan for any given problem. In practice, the necessary computer requirements for large problems are not realistically available. As a result, a new problem that can utilize available computer resources is created from the original problem by introducing the additional constraints relating to boundary specifications. However, the solution to this new problem is not guaranteed to be the optimal plan for the original problem. While there is no guarantee of optimality with respect to the original problem, the solution to the new problem may be very good or possibly even optimal to the original problem. Also, the solution to the new problem is computationally easier to obtain than the solution to the original problem, especially in the case of large problems.

2.3 THE PHASE I MODEL

The first phase of the algorithm enumerates all feasible districts within the state. To be feasible, the districts generated by the first phase of the algorithm must satisfy specified conditions for certain criteria, including population equality, contiguity, geographic compactness, and population unit grouping, as defined below.

Population Equality

Garfinkel justifies the use of population equality as a constraint in the development of his model in the following way.

The vast majority of districting plans which have been ruled unconstitutional failed on the grounds that there existed some district which had a disproportionately large or small population. As a result of these rulings, it is generally possible to get an indication of the maximum deviation allowed by the courts from the mean district population. Consequently, a constraint may be determined which says that no district population may deviate from the mean by more than a given number. (Garfinkel and Nemhauser, 1970, p. B-495)

As a result, he says that a district is feasible only if

$$(1) \quad |P(j) - \bar{p}| \leq \alpha \times \bar{p}$$

where

$P(j)$ = the population of district j ,

$\bar{p}$ = the mean district population,

α = the representation of the maximum allowable percentage deviation of the population of a district from the mean district population ($0 \leq \alpha \leq 1$).

Contiguity

The second criterion which must be satisfied in the first phase of the algorithm is the contiguity of the districts. A district is feasible only if it is contiguous. Garfinkel rigorously defines contiguity as follows. "Let $B = \{b_{ik}\}$ be a symmetric $N \times N$ matrix with

$$b_{ik} = \begin{cases} 0 & \text{if units } i \text{ and } k \text{ have a common boundary} \\ & \text{greater than a point (are contiguous)} \\ 1 & \text{otherwise.} \end{cases}$$

Consider a district to be an undirected graph, whose vertices are the units of the district, and an arc exists between vertices i and k if and only if $b_{ik} = 1$. The district is contiguous if and only if the graph is connected (a path exists between every pair of vertices)." (Garfinkel and Nemhauser, 1970, p. B-496)

(2) A district is feasible only if it is contiguous.

Compactness

The model considers only geographical compactness in its formulation, ignoring other definitions of compactness. A district that is considered by the model to be geographically compact is one with a generally circular or square shape as opposed to a long, thin shape. Garfinkel develops two compactness constraints, a "distance" compactness constraint and a "shape" compactness constraint. Shape compactness is represented as a dimensionless measure as described below. For feasibility, a district must have a shape compactness measure less than some input value. The distance compactness constraint does not allow two units to be in the same district if they are excessively far apart. Developed primarily as a computational device, the distance compactness constraint described below prevents the algorithm from generating districts which would not be shape compact anyway. Use of this constraint is critical to the solution of large problems.

Distance Compactness

The distance compactness constraint in the first phase of the algorithm states that a district is feasible only if

$$(3) \quad d(i,k) > e(i,k) \rightarrow a_{ij} \cdot a_{kj} = 0.$$

where

$d(i,k)$ = the distance between the approximate centers of units i and k ,

$e(i,k)$ = the exclusion distance for units i and k .

In other words, two units are not allowed to be in the same district if they are sufficiently far apart. Alternative implementation strategies include identifying a constant exclusion distance for all units in the state or allowing the exclusion distance to be a function of the relevant characteristics of the sectors of the state in which units i and k are located. (Garfinkel, 1968, p. 14)

Shape Compactness

The other compactness constraint for feasibility in the first phase of the algorithm is the shape compactness of a district. The model says that district j is feasible only if

$$(4) \quad c'_j \leq \beta$$

where

β = measure of shape compactness specified by the user ($0 \leq \beta < \infty$),

c'_j = a dimensionless measure of the shape compactness of district j

$$= d_j^2 / A(j)$$

where d_j = the distance between the approximate centers
of the units of j which are farthest apart,

$A(j)$ = the area of district j .

As c'_j decreases, district j is said to be more compact. (Garfinkel and Nemhauser, 1970, p. B-496)

Population Unit Grouping

Serving as a preprocessor to the execution of the Phase I algorithm, the interactive graphics routine produces population unit grouping information. During execution of the routine, certain population unit boundaries may be selected as included or excluded. A boundary that is selected as included means that the two population units that share that boundary *cannot* be in the same district. Likewise, a boundary that is selected as excluded means that the two population units that share that boundary *must* be in the same district.

Let $F = \{f_{ik}\}$ be a symmetric $N \times N$ matrix with

$$f_{ik} = \begin{cases} 1 & \text{if units } i \text{ and } k \text{ share a boundary that must be included} \\ & \text{in the final solution,} \\ 0 & \text{otherwise.} \end{cases}$$

A district is feasible if and only if

$$(5) \quad f_{ik} = 1 \rightarrow a_{ij} \cdot a_{kj} = 0$$

Let $G = \{g_{ik}\}$ be a symmetric $N \times N$ matrix with

$$g_{ik} = \begin{cases} 1 & \text{if units } i \text{ and } k \text{ share a boundary that must be excluded} \\ & \text{in the final solution,} \\ 0 & \text{otherwise.} \end{cases}$$

A district is feasible only if

$$(6) \quad g_{ik} = 1 \rightarrow a_{ij} \cdot a_{kj} = 1$$

In the first phase of the algorithm, a district is feasible only if it satisfies the conditions for population equality, contiguity, distance and shape compactness, and population unit grouping as described above.

2.4 THE PHASE II MODEL

Let $c_j = |P(j) - \bar{p}| / \alpha \times \bar{p}$ and assume that the $(N \times S)$ matrix A of feasible districts has already been determined in Phase I. The objective in Phase II is to find those plans (binary S -vectors X) which:

$$(7) \quad \text{minimize } \max_{j=1, \dots, S} c_j x_j$$

subject to

$$(8) \quad \sum_{j=1}^S a_{ij} x_j = 1, \quad i=1, \dots, N$$

$$(9) \quad \sum_{j=1}^S x_j = M$$

$$(10) \quad x_j = 0, 1, \quad j=1, \dots, S$$

The objective function seeks to minimize the maximum population deviation of any district in the plan. Constraint (8) says that every unit must appear in one and only one district of a plan. Constraint (9) specifies that there shall be exactly M representatives.

2.5 THE PHASE I ALGORITHM

The algorithm used to find all districts satisfying constraints (1)-(6) is a "tree search" algorithm. The algorithm begins at an arbitrary unit and adds on contiguous units until the combined population becomes feasible. If the district satisfies the population unit grouping constraint and compactness constraints, it is filed. Backtracking on the enumeration tree is done when the combined population exceeds the upper limit. During execution of the algorithm, periodic checks are made for enclaves, small groups of units isolated from the rest of the state by the proposed district. To reduce the time required for solution, checks for uncovered units are performed using a theorem developed by Garfinkel. For a discussion of the specifics of the original algorithm see (Garfinkel, 1968).

After the exclusion distances have been determined, an exclusion matrix Z is generated where

$$z_{ik} = \begin{cases} 1 & \text{if } d(i,k) > e(i,k) \text{ or } f_{ik} = 1 \text{ (i and k may not be in same district)} \\ 0 & \text{otherwise.} \end{cases}$$

The matrix Z may be modified in the course of the algorithm.

A "partial solution" to Phase I is a binary N-vector a_j which satisfies

$$P(j) \leq (1 + \alpha) \times \bar{p}$$

The units for which $a_{ij} = 1$ are contiguous

$$z_{ik} = 1 \rightarrow a_{ij} \cdot a_{kj} = 0$$

where the set of units for which $a_{ij} = 1$ represents a subset S_j of a potential district and $P(j)$ is the population of S_j . At any given time in the first phase the units are partitionable into the following classes:

- (1) Included units: units included in the current partial solution.
- (2) Excluded units: units excluded from the current partial solution.
- (3) Free units: units under consideration for inclusion with the units of the current partial solution.

Units tested for inclusion in the current partial solution are represented by a vector V , with units appearing in order of testing. If unit k is included it is written as k , if excluded it is written as $-k$.

The units are further partitioned into:

- (a) Non-tested units: units not appearing in V .
- (b) Once-tested units: if $V = (a, b, \dots, c, k, \dots)$, so that k is included, then the vector $V = (a, b, \dots, c, -k, \dots)$ still remains to be tried.
- (c) Twice-tested units: if $V = (a, b, \dots, c, k, \dots)$ and $V = (a, b, \dots, c, -k, \dots)$ has already been tried, then V is written as $(a, b, \dots, c, \underline{k}, \dots)$.

The following notation is used throughout the algorithm:

I : the set of included units.

$P(I)$: $\sum_{i \in I} p_i$

E : the set of excluded units.

S_j : the set of units in district j .

F : the set of free units.

$B(i)$: the set of units such that $b_{ik} = 1$.

$G(i)$: the set of units such that $g_{ik} = 1$.

$Z(i)$: the set of units such that $z_{ik} = 0$.

$B(S) = \bigcup_{i \in S} B(i)$, where S is an arbitrary set.

$$G(S) = \bigcup_{i \in S} G(i).$$

$$Z(S) = \bigcup_{i \in S} Z(i).$$

C : the set of free units contiguous to the included units.

N lists in which to file the generated districts. District j is filed in list i if i is the lowest numbered unit in district j . The districts are filed in order of increasing cost with ties broken arbitrarily.

A set of contiguous units $S \subset \bar{I}$ forms an enclave if $P(S) = \sum_{i \in S} p_i < (1 - \alpha) \times \bar{p}$

and there do not exist units $i \in S$ and $k \in (\bar{I} \cap \bar{S})$ such that $b_{ik} = 1$. An integer EN is used as an input to the algorithm to indicate how often an enclave check should be performed. Any district which causes an enclave cannot be in a plan.

The steps of the algorithm are as follows.

Step I (Initialization). Set $V = (1)$, $j = 0$ (j counts the districts generated), $e = 0$ (when $e = EN$ an enclave check is performed), $D = \emptyset$ (the set of "fixed" districts), $U = \emptyset$ (the units of D).

Step II (Unit annexation). If $e \geq EN$ go to Step IV. Otherwise find the lowest numbered unit i such that $i \in C$ and $P(I) + p_i \leq (1 + \alpha) \times \bar{p}$. If no such unit exists, go to Step V. Otherwise set $V = (V, i)$, $e = e + 1$. If $P(I) \geq (1 - \alpha) \times \bar{p}$, go to Step III. Otherwise, repeat Step II.

Step III (File district). If there is any unit $i \in I$ such that $g_{ik} = 1$ and $k \in \bar{I}$, go to Step II. Otherwise, compute c' for the units of I . If $c' > \beta$ go to Step II. Otherwise, let $j = j + 1$. District j is a feasible district. Let $c_j = |P(I) - \bar{p}| / \alpha \times \bar{p}$. Let i be the lowest numbered unit of I . File district j in list i in order of nondecreasing cost (c_j). Go to Step II.

Step IV (Enclave check). Check to see if an enclave exists. If there is no enclaved set S , set $e = 0$ and go to Step II. If set S forms an enclave and if there exists a unit i such that $i \in (S \cap E)$, or if any unit of S is excluded from any other

unit of S or if $P(I) + P(S) > (1 + \alpha) \times \bar{p}$, set $e = EN - 1$ and go to Step V. Otherwise set $e = 0$, let $S = (s_1, s_2, \dots, s_p)$ and set $V = (V, \underline{s_1}, \underline{s_2}, \dots, \underline{s_p})$. Then, if $P(I) \geq (1 - \alpha) \times \bar{p}$ go to Step III, but if $P(I) < (1 - \alpha) \times \bar{p}$ go to Step II.

Step V (Backtrack). Let i be the rightmost unit of V which is not underlined. Replace i by $\underline{i}$ and remove all entries to the right of i . If V contains N underlined entries terminate. Otherwise, set $e = e + 1$. If any entries of V are not underlined go to Step II. Otherwise, go to Step VI.

Step VI (Reduction). Let t be the rightmost unit of V . If $r_t = \emptyset$ then terminate since unit t is uncovered. Otherwise, find the lowest numbered unit i such that i is not in V . Let $C = (i)$. Go to Step II.

2.6 THE PHASE II ALGORITHM

In the second phase, the algorithm is also of the tree search variety. It is very fast, chiefly because the organization of the districts into lists allows the algorithm to be very effective. Specifically, it permits the algorithm to skip among lists, ignoring the districts that are not in the list of interest and thereby reducing the number of feasible districts the algorithm must examine. The algorithm is described in detail below.

A "partial solution" in the second phase is a vector X satisfying:

$$\sum_{j=1}^S a_{ij}x_j \leq 1, \quad i=1, \dots, N$$

$$\sum_{j=1}^S x_j \leq M,$$

$$x_j = 0, 1.$$

In the second phase, the variables x_j may be partitioned into the following classes:

- (1) fixed variables: those districts which must be in any plan,
- (2) trial positive variables: districts which are included in the current partial solution but which are not required to remain in later solutions,
- (3) trial zero variables: districts which are excluded from the current partial solution but which may appear in later solutions,
- (4) free variables: districts which are under consideration for inclusion in the current partial solution.

At the completion of Phase I, S districts (column vectors) have been generated. In addition each district j has been placed in one of N lists, where district j is in list i if the lowest numbered unit in the district is unit i . The districts have been filed in each list in order of nondecreasing cost (c_j) with ties broken arbitrarily. The set of fixed districts is D and the set of units covered by these districts is U . The number of districts in D is $N(D)$ and in Phase II, $M - N(D)$ districts which cover the units of $\bar{U}$ must be found. If $N(D) = M$ then Phase II is not entered.

The following notation is utilized in the second phase of the algorithm:

$f : 0 \leq f \leq M - N(D)$, the number of trial positive districts in the current partial solution,

Y : the set of districts in the current partial solution,

T : the units in the districts of Y ,

S_j : the set of units in district j .

Also of the tree search variety, the Phase II algorithm finds every optimal solution to the problem (7)-(10). The steps of the second phase algorithm are described below.

Step I (Initialization). Set $f = 0$, $Y = D$, and $T = U$. Go to Step II.

Step II (Choose next list). Find the lowest numbered unit i such that $i \in \bar{T}$. If there does not exist such a unit then all the units have been covered with fewer than M districts. Go to Step IV. If an uncovered unit is found, an indicator is set pointing to the "top" (lowest cost district) of list i . Go to Step III.

Step III (District Annexation). Begin at the indicated position in list i and examine in order of increasing cost the districts of the list. If district j is found such that $T \cap S_j = \emptyset$ then set $Y = Y \cup (j)$, $T = T \cup S_j$ and go to Step V. If no such district is found go to Step IV.

Step IV (Backtrack). There are no other plans containing the districts in the current partial solution (the districts whose units make up the set T). If $f = 0$ there are no other solutions and the algorithm terminates. Otherwise set $f = f - 1$. Subtract the units of the last trial positive district j in the partial solution from T . Set $Y = Y - (j)$. Set an indicator at the position in the list from which this last trial positive district was taken. Go to Step III.

Step V (Test for plan). Set $f = f + 1$. If $f = M - N(D)$ and the districts in the solution cover all the units, go to Step VI. If $f = M - N(D)$ and the districts in the solution do not cover all the units go to Step IV. If $f \neq M - N(D)$ go to Step II.

Step VI (Plan is found). A plan has been found whose value V is the largest c_j of the districts j in Y . Delete from each list any district n for which $c_n > V$. File the current plan as the incumbent. Go to Step IV.

2.7 CONCLUSION

The two phase algorithm described above finds all optimal solutions, for a given set of criteria, to political districting problems. The algorithm generates the set of all feasible districts and selects a subset of feasible districts from that set to cover all population units exactly once, while minimizing the maximum deviation of any district population from the mean district population.

The original algorithm was implemented in FORTRAN IV and assembly language on an IBM 7094. Use of the assembly language for the IBM 7094 resulted in efficient run times for problem solution, but also limited the transportability of the original implementation. Computational success was achieved without the use of exclusions for smaller problems ($N < 30$) and with exclusions for slightly larger problems ($30 < N < 40$). However, for very large problems ($N > 50$), no successful runs were completed. For a detailed account of the computational results originally achieved, see Garfinkel (1968).

The objectives of the current algorithm and implementation were threefold: increased portability, ease of use, and the solution of large problems. The current implementation uses only the higher level languages of Fortran and C to meet the objective of increased portability. The interactive graphics routines, mentioned earlier and described in detail in the next chapter, are included to meet the other objectives of ease of use and the solution of large problems. The overall resulting interactive graphical political districting system is described in the following chapter and solutions to real problems are given in the final chapter.

Chapter 3

THE SYSTEM

3.1 INTRODUCTION

This chapter describes the development and operation of the different components of the interactive graphical political districting system (IGPDS). First, a general overview of the entire system, its development, and its flow is given. The remainder of the chapter describes in detail the components and use of the system, including the programs that implement the algorithms presented in Chapter 2, the graphics logic and routines, and all datasets, both input and system generated.

As previously mentioned, the original objectives for the development of the IGPDS were threefold: increased portability, ease of use, and solution of large problems. The implementation of the original system included the use of Fortran and MAP, assembly language for the IBM 7094. While the use of assembly language allowed very efficient execution of the routines, its use severely limited the portability of the system. In contrast, the implementation of the new system utilized only standard versions of the higher level languages Fortran and C. While the use of higher level languages may reduce the efficiency of execution somewhat, the system meets the objective of transportability.

The second objective was to develop a system that is easy and "fun" to use and understand. In an effort to achieve this objective, the system developed is an entirely menu-driven system that also allows graphical representations of solutions. By providing the user with explanatory prompts, a menu-driven system is

inherently easier to use than a system without menus. As discussed in Chapter 1, a graphical representation of solution data greatly increases the quantity and quality of human understanding of that data.

However, the utilization of graphics in the system is not strictly to meet the objective of developing an easy and "fun" system to use. The graphics play a much more important role as a computational aid. As is typical of enumeration algorithms, the first phase of the algorithm discussed in Chapter 2 tends to grow exponentially and limiting this growth, particularly in the solution of large problems, is crucial. Attesting to this fact is that the computational success of the original implementation was limited to small to medium sized problems ($N < 40$). As discussed earlier, part of this limitation was due to the size and speed of the computer, but undoubtedly the exponential growth was also a major contributor to this limit.

Through the graphics routines, the user may specify that certain population unit boundaries must either be included or excluded as district boundaries in the final solution. These selections have the effect of limiting the possible combinations of population units in a districting plan. The boundary specifications prior to execution of the first phase of the algorithm attempt to limit the potential growth of the problem by effectively reducing larger problems into smaller ones. In other words, without the boundary specifications, the algorithm enumerates *all* feasible districts with regard to population equality, contiguity, and compactness. The boundary specifications add the additional constraint of certain population unit groupings and, as a result of this constraint, there are fewer feasible districts for the algorithm to find.

A side benefit to the use of the graphics routines is the ability of the system to recognize and maintain any existing boundaries, such as natural or ethnic boundaries, that are deemed important by the user. For example, the user of the

system may decide that population units on either side of a mountain range should not be part of the same political district. Including the boundary separating these population units during the graphics routine prior to the solution of the problem achieves the user's objective, since units sharing this boundary will not be in the same district in the final plan.

The IGPDS was developed and implemented under the VMS operating system on a VAX 1170 at the University of Tennessee, Knoxville. The system consists of a DCL shell, five main processes, written in the higher level languages of Fortran and C, and sixteen data files, which are described in greater detail below. Graphics portions of the system utilize the commercially available DISSPLA graphics software and Terminal Control System (TCS) and were written to be viewed on a Tecktronix model 4014 (or later) or Hewlett-Packard model 7550 plotter.

The next section gives a general overview of the IGPDS. The following two sections contain detailed descriptions for each of the programs and datasets that comprise the system. Finally, step by step instructions for running the system are given.

3.2 GENERAL FLOW OF THE IGPDS

Figure 3-1 shows a schematic flow diagram for the interactive graphical political districting system. The Fortran program PREPRO.FOR serves as a preprocessor to a model run by allowing the user to create or modify the data files necessary for a run. Represented on the flow diagram by the disk labelled DATA, output files from this program include the files DATA.DAT, CONTIG.DAT, DIST.DAT, EXCLUDE.DAT, and PARAM.DAT. With the exception of the file PARAM.DAT, the information stored in these files is the static information about the state or the area to be districted, including population, contiguity, and distance

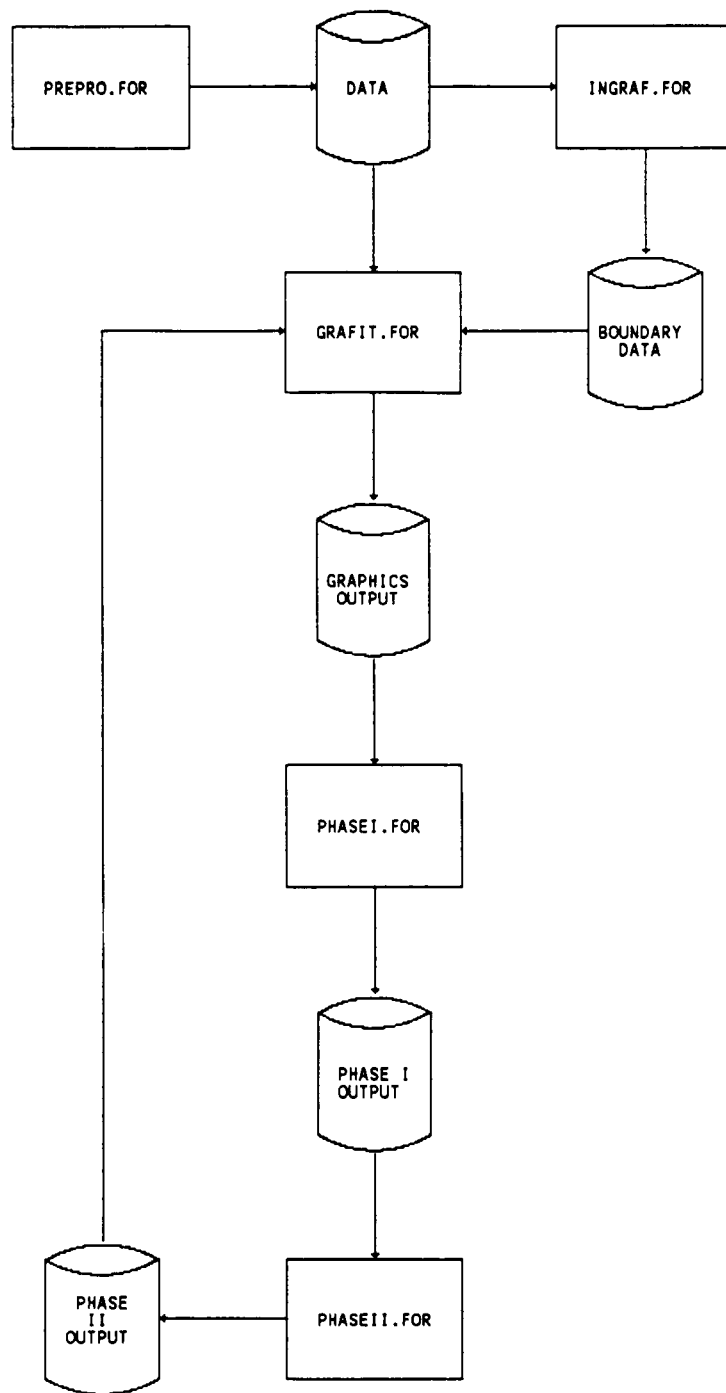


Figure 3-1: Schematic Flow Diagram for the Interactive Graphics Political Districting System (IGPDS).

information. The file PARAM.DAT contains parameter values for each run, so these values may change from one run to the next. The information in the data files created or modified by PREPRO.FOR is utilized by three programs, INGRAF.FOR, GRAFIT.FOR, and PHASEI.FOR.

For each state or area to be districted, the graphics routine INGRAF.FOR is run one time only prior to any model runs. The program simply reads in contiguity information about the state or area and prompts the user to digitize points for each boundary, which are then stored in the file BOUND.DAT.

The other graphics program GRAFIT.FOR reads in the file BOUND.DAT to display a map of the state or area under consideration. Additionally, any solution information from a previous run of the model is read from the file PHASEII.OUT and the resulting districting plan is displayed. Finally, contiguity information for the state is input and used to create the population unit grouping files GRAFIN.OUT and GRAFEX.OUT resulting from any district boundary selections made by the user during execution of GRAFIT.FOR.

Corresponding to the first phase of the algorithm described in Chapter 2, the program PHASEI.FOR reads in population, contiguity, and distance information output from PREPRO.FOR and population unit grouping information output from GRAFIT.FOR. The program uses that data to generate all feasible districts for the state or area under consideration and stores the resulting output in the files PHASEI.OUT, IA.OUT, LA.OUT, and LIST.OUT.

The files output from PHASEI.FOR are read into the program PHASEII.FOR, which corresponds to the second phase of the algorithm described in Chapter 2. The information describing the optimal districting plan(s) found during execution of this program is written to the file PHASEII.OUT, which can then be read by the graphics routine GRAFIT.FOR to display the solution found during this model run.

3.3 PROGRAM DESCRIPTIONS

The following section describes in greater detail the programs of the IGPDS, including the general purpose of the program, inputs, outputs, and a discussion of any specific features of the program. Detailed descriptions for the datasets referred to in the program descriptions may be found in Section 3.4.

A. PREPRO.FOR

Serving as a preprocessor to the system, this utility program simply automates the creation and modification of the data files necessary for running the overall system. This program should be used to create a set of data files for each state or area for which a districting plan is to be found. At the start of execution of this program, the user inputs a two character code which will serve to uniquely identify the set of data files about to be created. With the exception of one file, all the data files generated by PREPRO.FOR contain generally unchanging information about the state or area to be districted (such as contiguity and distance information), so these static files need be created only once and stored as unique datasets. They can then be loaded into the system for a run using the unique two character code assigned by the user.

All information input into this program is entered by the user via the keyboard. All requests for information within this menu driven program are accompanied by an explanatory prompt.

Six files may be created or modified using this program. The file ID.LOG, a log file listing the unique codes identifying the sets of files created, is updated each time this program is run. The file PARAM.DAT contains the parameters for the current model run. These parameters may be modified by the user during execution of PREPRO.FOR prior to each complete run through the system. The static information about the state or area, including population, contiguity, and

distance information, is stored in the files CONTIG.XX, DATA.XX, DIST.XX, and EXCLUDE.XX, where XX is the unique two character code identifying this set of data files. These files contain all the necessary information in the appropriate format for running the system.

B. INGRAF.FOR

This Fortran program, which is linked with the commercially available graphics package DISSPLA, allows the user of the system to digitize and store the map of any area or state that is to be districted. As with the static information generated by PREPRO.FOR, the map file created by INGRAF.FOR need be created only once and stored using the unique two character code specified by the user.

INGRAF.FOR reads in the contiguity data file for the state or area to be digitized and, for all pairs of population units that are contiguous, prompts the user to input boundary points via a mouse and a digitizing tablet. Additionally, the user is prompted to enter boundary points for each external boundary. For each boundary digitized, as many points may be entered as necessary to accurately define the boundary.

Two files are generated during execution of this program, BOUND.XX and LIMITS.XX where XX is the unique two character code specified by the user to identify these files. BOUND.XX contains all the boundary points digitized by the user during execution of INGRAF.FOR. The file LIMITS.XX contains the (X,Y) coordinates that define the window to best display the digitized map on the graphics terminal or plotter. Both files output from this program serve as necessary input files to the other graphics program, GRAFIT.FOR.

C. GRAFIT.FOR

Also a Fortran program linked with the commercially available graphics software DISSPLA, GRAFIT.FOR has two major functions. First, the program

shows the current solution by displaying the state or other area under consideration and highlighting the district boundaries for the most recent districting plan found by the system. The second function is to allow the user to prepare population unit grouping information for another run of the system by selecting boundaries that must be included or excluded in the next plan found by the system. In other words, this program displays the most recent plan found by the system and allows additional constraints to be placed on the problem before the system finds another districting plan.

Input for this program comes from three different sources. The first source is the output files from INGRAF.FOR, BOUND.XX and LIMITS.XX, where XX is the two character code uniquely identifying the datasets of the state or other area under consideration. These files contain the digitized map points and an appropriate window definition for viewing the map on the graphics terminal or plotter. Also input to this program is the contiguity information CONTIG.XX generated by the program PREPRO.FOR. The final input source for GRAFIT.FOR is the file containing the information defining the districting plan found in the most recent run. That file is called PHASEII.OUT and is output from the program PHASEII.FOR, which is described below.

After all the input data has been read, the program draws the map of the state on the screen or plotter, displaying the current districting plan by highlighting all the district boundaries in the plan. If the output is directed to a graphics terminal, the program enters the interactive routine in which the user is allowed to specify boundary constraints for the next plan found by the system.

The first option available in the interactive routine is to fix a boundary in the next plan. In other words, that boundary must exist as a district boundary in the next plan found by the system. Therefore, the pair of population units that share that boundary cannot exist in the same district in the final plan. Likewise,

another option is available that removes a boundary from the next plan. In the next plan found by the system, that boundary cannot exist as a district boundary and the pair of population units that share that boundary must be in the same district. The program allows as many boundaries specifications as desired.

There are three main outputs of the program GRAFIT.FOR. First is the graphical display on the screen or plotter of the current districting plan. The second output from this program consists of two files, GRAFEX.OUT and GRAFIN.OUT, containing population unit grouping information, based on the user's selected boundary constraints. The file GRAFEX.OUT contains a list of pairs of population units that are excluded from existing in the same district in the next plan found by the system. GRAFIN.OUT contains a list of pairs of population units that must exist in the same district in the next plan found by the system. Both of these files are accessed during execution of PHASEI.FOR, which is described below. The final type of output is a log file called BOUND.LOG which stores a list of boundaries selected as included or excluded during execution of the interactive routine. This file may be accessed later as a reminder of the actions taken during the most recent run of GRAFIT.FOR.

D. PHASEI.FOR

In general, the purpose of this program is to enumerate all feasible districts for a state or other area, where feasibility indicates meeting specified standards for contiguity, compactness, population deviation, and population unit grouping.

Found in the file PARAM.DAT, constants input into this program consist of the number of required districts, the maximum allowable percentage deviation of a district from the mean district population, the maximum allowable measure of shape compactness, and an enclave check counter. The other files generated by PREPRO.FOR provide PHASEI.FOR with information for each population unit in

the state or area. The file DATA.XX provides population and area for each population unit. Contiguity information is found in the file CONTIG.XX while the file DIST.XX contains the distances between each pair of population units. For each population unit, the file EXCLUDE.XX lists all the population units that lie beyond the current exclusion distance.

The final source of input for PHASEI.FOR is the population unit grouping information generated during execution of GRAFIT.FOR and stored in the files GRAFEX.OUT and GRAFIN.OUT.

The program PHASEI.FOR consists of six Fortran subroutines, corresponding to the six steps of the Phase I algorithm described in Chapter 2. Also included in this program is the subroutine SORT.C, which takes the lists of districts generated during the course of the algorithm and sorts them in order of nondecreasing cost. PHASEI.FOR enumerates all feasible districts for an area or state and writes them to four output files, PHASEI.OUT, IA.OUT, LA.OUT, and LIST.OUT. These four files contain information describing each feasible district and its associated cost.

E. PHASEII.FOR

This program accesses the four files describing all the feasible districts generated in PHASEI.FOR and finds the set of districts which covers each population unit exactly once. The program chooses the set so that it minimizes the maximum deviation of district population from the mean district population.

Information input into this program consists of information generated during execution of PHASEI.FOR. PHASEI.OUT provides the number of required districts and the number of feasible districts generated in the first phase execution. The number of units in each feasible district is found in the file IA.OUT while LA.OUT contains the lists of population units in each district. LIST.OUT contains the sorted lists and costs of the feasible districts.

The program PHASEII.FOR consists of six Fortran subroutines corresponding to the steps of the Phase II algorithm described in Chapter 2. The output of this program, stored in the file PHASEII.OUT, is the optimal districting plan(s) found. The file PHASEII.OUT contains the number of plans found, the optimal value and the districts that comprise the optimal plan(s). Additionally, the current solution may be saved into another user-specified file for later recall and display.

F. GO.COM

This DCL program serves as the driver for the remainder of the IGPDS. It consists primarily of a menu system to allow the user to run desired programs and perform some utility functions. These utility functions include actions such as loading the appropriate data files as selected by the user, confirming the existence of the files necessary for a run, and listing log files that may be generated during execution of one of the system programs.

3.4 DATA FILES

The following section describes the entire set of data files used in the Interactive Graphical Political Districting System, including input files, output files and system generated files. Included for each file is a discussion of the purpose of the data in the file, the format of the data file, the creating program, and any routines that access the file.

A. BOUND.DAT

Containing all the digitized points input by the user via INGRAF.FOR, this file is accessed by GRAFIT.FOR for use in map drawing. The information in the file alternates between the number of points input by the user to define a boundary between two contiguous units and the actual X and Y coordinates digitized by the user. A representation of the format of the file is given.

```

NP(1,1)
PX(1,1,1) PY(1,1,1) .... PX(1,1,w) PY(1,1,w)
NP(1,2)
PX(1,2,1) PY(1,2,1) .... PX(1,2,x) PY(1,2,x)
:
NP(i,j)
PX(i,j,1) PY(i,j,1) .... PX(i,j,y) PY(i,j,y)
:
NP(N,N)
PX(N,N,1) PY(N,N,1) .... PX(N,N,z) PY(N,N,z)

```

where

NP(a,b) = the number of points input to define the boundary between units a and b. If a = b, the points input represent the external boundary of unit a.

PX(i,j,k) = the kth X coordinate of the boundary between units i and j.

PY(i,j,k) = the kth Y coordinate of the boundary between units i and j.

N = the number of units in the area to be districted.

w,x,y,z = the number of points input to define a boundary.

B. BOUND.LOG

This file contains a log generated by GRAFIT.FOR that shows the boundaries selected by the user for inclusion or exclusion. Also, information is displayed if the user did not highlight a defined boundary. The user has the option from the main driver routine GO.COM to list this file to determine what actions were taken.

C. CONTIG.DAT

Created by the program PREPRO.FOR, this file consists of the number of units in the area to be districted, an indicator of the number of units contiguous to each unit, and a list of units contiguous to other units. The information in this file is accessed by the program PHASEI.FOR for execution as well as the two graphics routines, INGRAF.FOR and GRAFIT.FOR, for use in prompting for

digitized points, map drawing, and boundary identification. Following is a representation of the file format.

N

IA(1) IA(2) IA(i) IA(N)

LA(1) LA(2) LA(i) LA(y)

where

N = the number of units in the area to be districted

IA(i) = the number of units contiguous to unit i

LA = the list of all contiguous units

(LA(1) through LA(IA(1)) are units contiguous to unit 1)

D. DATA.DAT

This file contains the basic population and area information about the area to be districted for execution of the first phase of the algorithm. Accessed by PHASE1.FOR for execution, this file is either user created or generated by the program PREPRO.FOR and has the following format.

The first line(s) contains the population for each unit, with the position in the list of populations corresponding to the unit number. In other words, the first population corresponds to population unit one, the second to unit two, and so on. On the line(s) following the population information is a measure of area for each of the population units with the same format as the population data.

E. DIST.DAT

Also either user created or generated by the program PREPRO.FOR, this file contains geographical distance information for the population units in the area under consideration. The information consists of the distance, in miles, between the approximate geographic centers of all units. A representation of the format of the file follows.

D(1,2) D(1,3) D(1,j) D(1,N)

D(2,3) D(2,4) D(2,j) D(2,N)

:

D(i,i+1) D(i,j) ... D(i,N)

:

D(N-1,N)

where

D(i,j) = distance (in miles) between unit i and unit j

N = number of population units.

F. EXCLUDE.DAT

Generated by PREPRO.FOR or user created, this file contains lists of population units excluded from each other because the distance between them exceeds the user input exclusion distance. In PREPRO.FOR, the user is asked for an exclusion distance and the program creates exclusion indices and writes them to this file for use by PHASEI.FOR. The format of the file follows.

EXD

IE(1) IE(2) ... IE(N)

LE(1) LE(2) ... LE(y)

where

EXD = the user input exclusion distance

IE(i) = the number of population units excluded from unit i

LE = list of units excluded from other units,

(LE(1) through LE(IE(1)) are units excluded from unit 1)

N = the number of population units

y = total number of units excluded from other units.

G. GRAFEX.OUT

This file is one of two generated during execution of the graphics program GRAFIT.FOR. The information in this file consists of pairs of population units that may not co-exist in the same district in a plan. This list of population unit

pairs is determined during execution of the graphics program from the user's selections of boundaries that must be included as district boundaries in a plan.

H. GRAFIN.OUT

This file is the other file generated during execution of the graphics program GRAFIT.FOR. The information in this file consists of pairs of population units that must exist in the same district in a plan. This list of population unit pairs is determined during execution of the graphics program from the user's selections of boundaries that cannot be included as district boundaries in a plan.

I. IA.OUT

The information in this file is generated during execution of PHASEI.FOR and is used by the program PHASEII.FOR for finding the optimal plan. This file contains one record for each feasible district generated by PHASEI.FOR. The information stored in this file for each feasible district is the number of population units that comprise that district. This file is used as an index to the file LA.OUT described below to fully define each district generated in terms of the population units.

J. ID.LOG

The information in this log file is created during execution of PREPRO.FOR and is used for loading datasets for a new run of the system. The information in this file consists of a list of codes the user input to uniquely identify different datasets. When attempting to load new datasets to run the system, the user may display this list as a reminder of the codes of available datasets.

K. LA.OUT

The information in this file is generated during execution of PHASEI.FOR and is used by the program PHASEII.FOR for finding the optimal plan. This file contains one record of variable length for each of the feasible districts found

during execution of PHASEI.FOR. The variable record for a district stores the list of population units that comprise that feasible district. This file is used in conjunction with IA.OUT described above to reference the population unit information for each feasible district.

L. LIMITS.OUT

Output from the graphics program INGRAF.FOR, this file is accessed by the other graphics program GRAFIT.FOR. Stored in the file are the minimum and maximum X and Y coordinates input from the digitizing tablet during execution of INGRAF.FOR. This information is used by GRAFIT.FOR to automatically scale the area drawn so the entire map may be displayed on the graphics terminal or plotter.

M. LIST.OUT

The information in this file is generated during execution of PHASEI.FOR and is used by the program PHASEII.FOR for finding the optimal plan. This file contains three values for each of the feasible districts found during execution of PHASEI.FOR. The first value is an integer number representing the list in which this district is filed. Each district is filed in the list corresponding to its lowest numbered population unit. The second value for each district is the position of that district in the sorted list. The final value stored is the district's cost or value, which is the deviation of that district's population from the mean district population.

N. PARAM.DAT

This small file contains the run parameter information as entered by the user in the PREPRO.FOR program. The information in the file is stored on one line and consists of M, α , β and EN where

α = the maximum allowable population deviation,

β = the maximum allowable shape compactness,

EN = frequency indicator for enclave check,

M = number of required districts.

This file is accessed several times throughout a complete run of the system by PHASEI.FOR for execution and GRAFIT.FOR for displaying the information to the user.

O. PHASEI.OUT

The information in this small file is generated during execution of PHASEI.FOR and is used by the program PHASEII.FOR for finding the optimal plan. This file stores three values only, the number of units in the state or area under consideration, the number of required districts for that state or area, and the number of feasible districts generated during the most recent execution of PHASEI.FOR.

P. PHASEII.OUT

As the name implies, this file is generated during execution of the program PHASEII.FOR. The information stored in this file represents the optimal districting plan by this run through the system. The program GRAFIT.FOR accesses this file to graphically display the solution. The file stores the number of optimal plans found, the value of the plan(s), and, for each plan found, the districts that comprise the plan.

3.5 USE OF THE SYSTEM

The following sections provide information for the actual running of the Interactive Graphical Political Districting System, (IGPDS) serving as a basic user's manual to the system. Steps to take in running the system and options available to the user are discussed.

After logging on to the VAX, to enter the IGPDS, the user types GO at the prompt. A main menu for the system, like the one in Figure 3-2, is then presented.

The first step is to assure that the appropriate data for the area to be districted is stored in the proper format in the files DATA.DAT, CONTIG.DAT, DIST.DAT, EXCLUDE.DAT, and PARAM.DAT since the entire system is dependent on this information. This is accomplished by selecting the LOAD DATA FOR MODEL RUN option to load data an existing set of data files. If no data files exist for the state or area to be districted, the DATA FILE CREATION option to create new data files must be selected. Selection of this option puts the user into the PREPRO.FOR program, in which the user is asked for a two character code that will serve to uniquely identify the data files being created. The user is then presented with another menu that looks like the one in Figure3-3. Options are available to create all of the required datasets into the proper format. Upon selection of an option, the user is prompted to enter the desired information, for example, which units are contiguous to a certain unit. Before writing the input information to the appropriate file, the program asks the user for confirmation that the information entered is correct. In general, the user should create the files in the order they are presented by this data file creation menu. Upon exiting from this program, the user is returned to the main system menu.

Another preparation step is the confirmation of the contents of the boundary file BOUND.DAT. If the user desires to work with a new area or state, the option INPUT MAP POINTS is selected to digitize map points for that state or area. Before executing this program, the driver program asks for confirmation that the user is logged to a terminal connected to a digitizing tablet. This allows data manipulation and execution of the two phase algorithm to be accomplished without the use of a graphics terminal. During execution of the program INGRAF.FOR, the user is prompted to input points, using the mouse and the digitizing tablet, that

INTERACTIVE GRAPHICS POLITICAL DISTRICTING SYSTEM

MAIN MENU

<u>OPTION</u>	<u>ACTION</u>
1	DATA FILE CREATION / MODIFICATION
2	INPUT MAP POINTS
3	LOAD DATA FOR MODEL RUN
4	INTERACTIVE GRAPHICS / DISPLAY CURRENT SOLUTION
5	GENERATE ALL FEASIBLE DISTRICTS (PHASE I)
6	FIND THE OPTIMAL DISTRICTING PLAN (PHASE II)
7	QUIT

ENTER DESIRED OPTION:

Figure 3-2: Opening menu for the IGPDS.

completely define the boundary between two units. External boundaries are requested as the boundary between a unit and itself. The user is returned to the main system menu upon exiting from this program.

For a model run, in order to use any set of data files, either newly created or previously existing, the option LOAD DATA FOR MODEL RUN must be selected. A list of available datasets is displayed and the user must enter the two character code identifying the desired set of data files to load.

After ensuring that all the appropriate data and boundary files have been created and loaded into the system, the user is ready to begin a model run. The first step in a model run is selection of the INTERACTIVE GRAPHICS /

DATA FILE CREATION MENU

<u>OPTION</u>	<u>DATA FILE(S) CREATED</u>
P	PARAMETERS
A	AREA AND POPULATION
C	CONTIGUITY
D	DISTANCE
E	EXCLUSION
Q	QUIT

ENTER DESIRED OPTION:

Figure 3-3: Data file creation menu for the IGPDS.

DISPLAY CURRENT SOLUTION option, which displays, on the screen or plotter, a map of the state or area under consideration and any districting plan existing from a previous model run. As an example of what is typically displayed on a graphics terminal, Figure 3-4 shows the results for a sample model run for the state of Nevada. The map with the existing districting plan highlighted is displayed in the center of the screen with run parameters and the plan's value given on the right. Instructions and interactive options are listed on the left side of the screen.

Had the output been directed by the user to a plotter, none of this information would be displayed on the screen and no interactive options would be available to the user. However, if the output is directed to a graphics terminal, the user then has the interactive options available. Once the information has been displayed to the screen, the user may interact with the system using the cross hairs which appear on the screen.

Selection of an option is accomplished by moving the cross hairs to the desired option and hitting the space bar. The options available to the user are

INSTRUCTIONS.

HIGHLIGHT OPTION
WITH CURSOR.

THEN PRESS SPACE
BAR TO SELECT.

- A. FIX BOUNDARY.
- B. REMOVE BOUNDARY.
- C. RESTART.
- D. RESTORE ANOTHER
PLAN.
- E. QUIT.

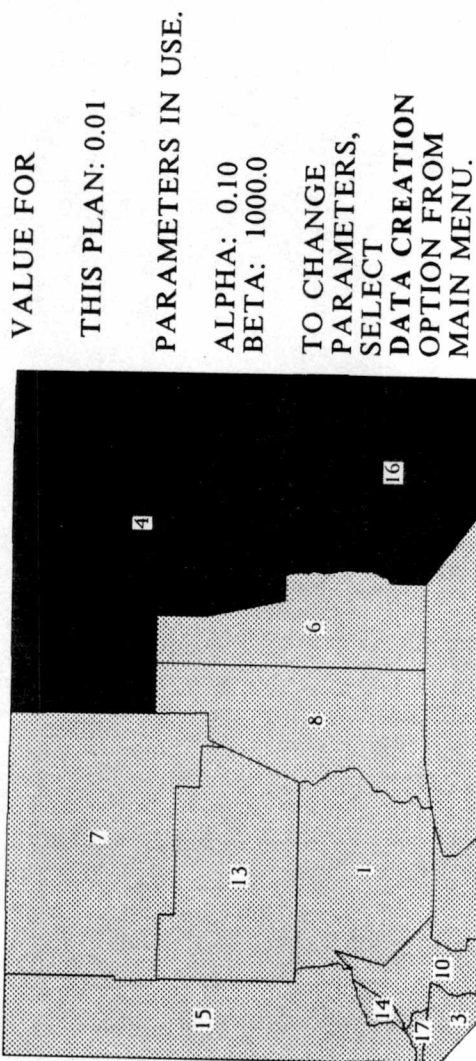


Figure 3-4: Sample graphics display and menu.

fixing a boundary, removing a boundary, restarting, restoring another solution, and quitting. Changes to the run parameters α and β are made by quitting the graphics routine and selecting the DATA FILE CREATION / MODIFICATION option.

Upon selecting either the option to fix a boundary or to remove a boundary, the user again has the cross hairs prompt. The program is waiting for two points which define a particular boundary. The user must position the cross hairs on a distinctive boundary point (i.e. the ends of the boundary) and hit the space bar. This procedure is repeated for the second boundary point. Should the user miss the boundary, a message indicating so is displayed at the bottom of the screen and the user may try again after selecting the same option from the menu. The cross hairs are still visible and are positioned so as to select the next option the user desires.

The option to restart is provided should the user desire to erase all boundary selections made so far during this session, and begin again. Additionally, the user may elect to restore a solution from a previous run. The name of this stored solution file must be provided by the user. Upon selecting the option to quit by hitting the space bar, the return key must be hit to exit from the graphics portion of the program. A slight delay follows while the program processes the information it has received and the system returns to the main system menu.

Next, the option GENERATE ALL FEASIBLE DISTRICTS which executes the program PHASEI.FOR is selected to generate the files of all feasible districts. Upon completion of the district enumeration, a message is output displaying the amount of time the program took to execute. Then the option FIND THE OPTIMAL DISTRICTING PLAN is selected to execute PHASEII.FOR and find the optimal districting plan(s) for the area or state. Again, the time for program execution is displayed upon completion of this program as well as the number of

optimal plans found. The user then has then option of saving the solution file under a different name to display it during some later model run.

After the optimal districting plan has been found, the option INTERACTIVE GRAPHICS / DISPLAY CURRENT SOLUTION is selected to graphically display the solution. From this point, the user may plot the solution, exit the system, or choose to include or exclude additional district boundaries and find another districting plan following the same procedure described above.

Chapter 4

RESULTS

4.1 INTRODUCTION

Four real problems of varying degrees of difficulty were solved using the Interactive Graphics Political Districting System (IGPDS). For each of these problems, the 1980 Census of the United States served as the source of input information. The existing districting plans and maps for Nevada, New Mexico, Washington, and West Virginia are described below as well as the plans found by the IGPDS system for each of the states. Also, a discussion comparing the current results with those from the original implementation of the algorithm is included. Finally, areas of improvement and enhancements to the system are discussed.

4.2 NEVADA

Nevada was selected as a problem with a low level of difficulty since it consists of small number of population units (18 units) and requires only two congressional districts. The population of Nevada is 800,493 divided among two districts results in a mean population of 400,246.5. The present solution, shown in Figure 4-1, has a maximum population deviation of 0.097% and a maximum measure of shape compactness of 1.4.

The heavy concentration of population in the southeastern corner of the state limits the number of "good" plans. As a matter of fact, the first run through the system reproduced exactly the current districting plan for Nevada. (See Figure 4-1.) The plan has population unit 18 existing as a district by itself. To form a

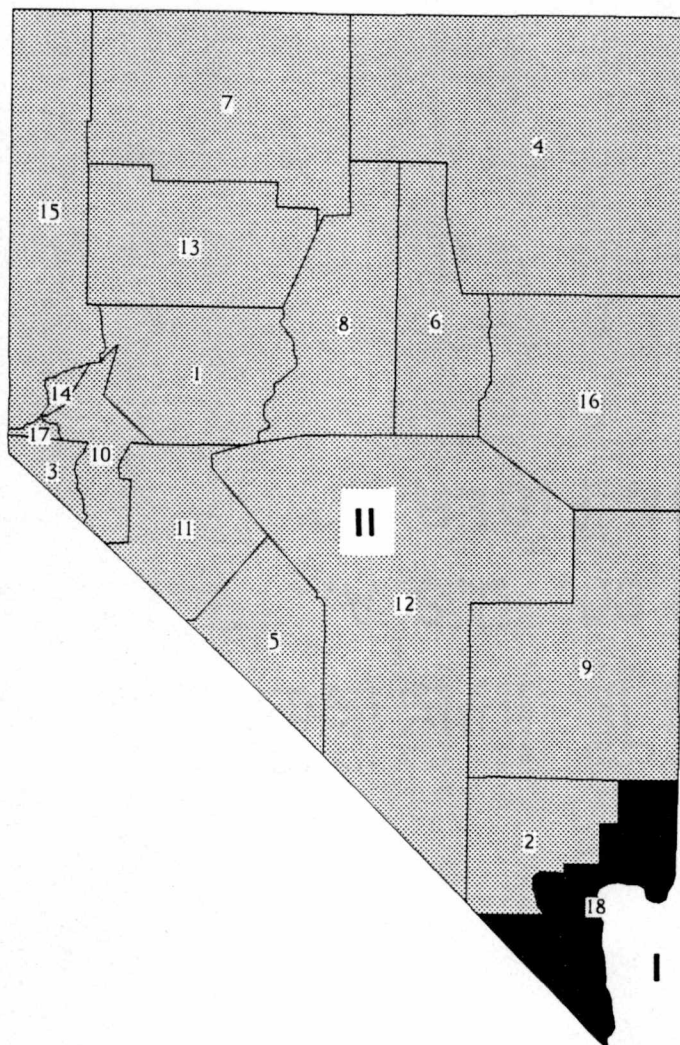


Figure 4-1: Nevada current plan.

different plan, another population unit must be included into the district with population unit 18. However, the population of unit 18 is 400,636, which is greater than the mean population of 400,246.5. Therefore, any other plan will have a worse population deviation than the current plan, since adding a unit's population to that of unit 18 causes the new district population to be even greater than the mean population. As a result, runs demonstrating the use of boundary exclusions and inclusions produce plans for Nevada that are not as good, in terms of population deviation, as the current plan.

Table 4-1 gives all the computational results obtained during the runs for Nevada described below.

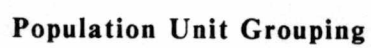
As previously mentioned, the first run through the system, with no boundary inclusions or exclusions, reproduced the existing plan for Nevada, as shown on the previous page. Since most of the interesting geography for the state is to the north and west of the population concentration and no solution better than the existing plan may be found, the boundary inclusions and exclusions in the following runs are not used for practical reasons, that is to "improve" the plan in some manner. Rather, they are used strictly to demonstrate the effects of the additional population unit grouping constraints on the optimal plan found by the system.

For example, in the second run, the boundary between population unit 18 and unit 9 is specified to be removed. This specification has the effect of saying that, for whatever reason, Lincoln county (unit 9) must be in the same district as that part of Clark county that is unit 18. As shown in Figure 4-2, the system found a plan that respects the population unit grouping specified. The population deviation for the new plan increased, as expected, to 1.03%, while the new maximum measure of shape compactness for the plan became 1.6.

The third run, for which results are shown in Figure 4-3, extends the population unit grouping constraint to include multiple groupings. As an example,

Table 4-1: Computational Results for Nevada.

	Run 1 (Figure 4-1)	Run 2 (Figure 4-2)	Run 3 (Figure 4-3)	Run 4 (Figure 4-4)
<u>INPUT</u>				
α	0.02	0.02	0.08	0.16
β	∞	∞	∞	∞
EN	∞	∞	∞	∞
N1	3	3	3	3
Exclusion distance	∞	∞	∞	∞
Boundary Exclusions	None	Figure 4-2	Figure 4-3	None
Boundary Inclusions	None	None	None	Figure 4-4
<u>OUTPUT</u>				
Districts generated	39	10	8	751
Phase I time	8.2 s	8.2 s	8.3 s	3.9 s
Phase II time	0.19 s	0.19 s	0.19 s	0.20 s
Maximum population deviation	0.00097	0.0103	0.0739	0.1570
Maximum shape compactness	1.4 (j=2)	1.6 (j=2)	3.1 (j=1)	1.0 (j=2)
Alternate optima	None	None	None	None



Boundary Inclusions

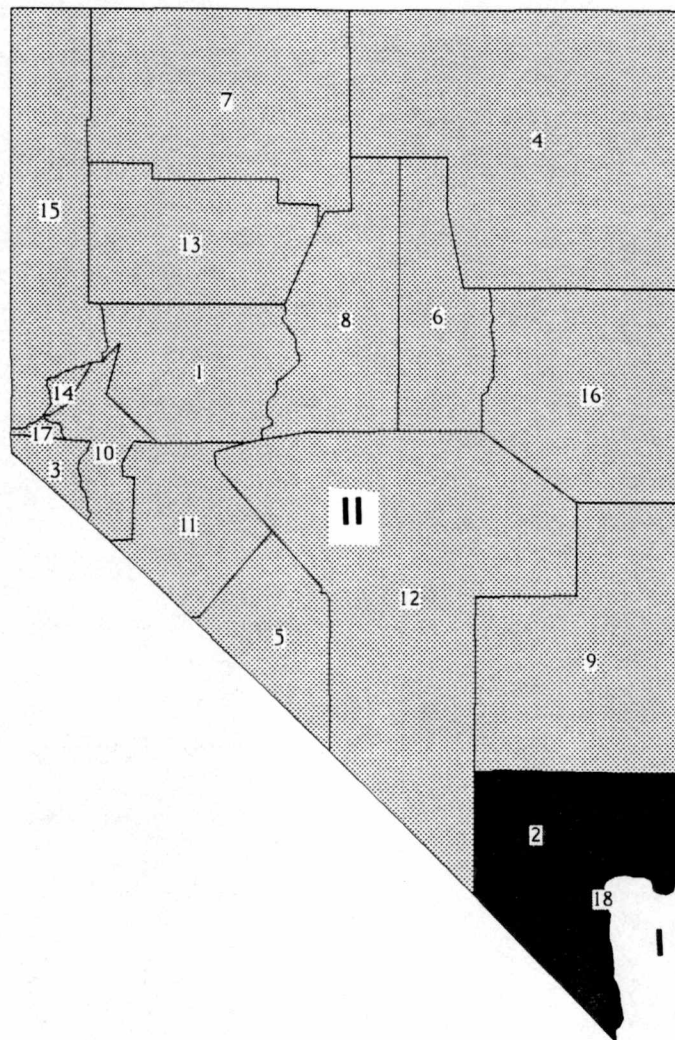
None.

Figure 4-3: Nevada - Run 3 results.

suppose all the population units that share a boundary with Utah must be in the same district (again, for whatever reason). In other words, unit 18 (part of Clark county) unit 9 (Lincoln county), unit 16 (White Pine county), and unit 4 (Elko county) must all be in the same district. Therefore, from the graphics routine, the boundaries selected to be removed from consideration as district boundaries are the boundary between units 18 and 9, the boundary between units 9 and 16, and the boundary between unit 16 and 4. To allow for the expected increase in population deviation, α was increased from 0.02 to 0.08. As shown in Figure 4-3, the first district in the new plan contains all the population units that border Utah, as desired. We note the population deviation increased to 7.39% and the new maximum measure of shape compactness for the plan worsened to 3.2.

The final run for Nevada demonstrates the effect of including a boundary, which prevents the two units that share that boundary from existing in the same district in a plan. All of the above grouping constraints are removed and the boundaries between units 2 and 9 and units 2 and 12 are fixed, that is, they must exist as district boundaries. This boundary inclusion has the effect of forcing units 2 and 18 to exist in the same district and because of the high population in these two units, α is increased to a generous 0.16. As a result of this increase, the number of feasible districts generated during execution of Phase I increased to 751 districts. The new plan, with a population deviation of 15.70% and maximum measure of shape compactness of 1.0, is displayed in Figure 4-4.

Since Nevada has few population units and a very uneven population distribution, these experiments are not necessarily practical. However, we note that the current districting plan for Nevada is optimal with respect to minimizing population deviation. Also, we are able to discern from the results some interesting trends that occur during the course of the solution process that continue to hold true for later experiments.



Population Unit Grouping

Boundary Exclusions

None.

Boundary Inclusions

2 & 9, 2 & 12

Figure 4-4: Nevada - Run 4 results.

We note from the results that as we tightened the population unit grouping constraint by specifying district boundaries, the final plan worsened with respect to population deviation, as expected. However, this behavior is not necessarily negative, since other factors, such as geography, may play important roles in determining a plan.

Another interesting observation is the faster execution time for Phase I in the fourth run, despite the generation of many more feasible districts than in the other runs. In the fourth run, sets of population units were excluded from existing in the same district, whereas in the earlier runs, sets of units were included into the same district. The algorithm does not check for violation of the constraint that says two units must be in the same district until after a district feasible with respect to population, contiguity, and compactness has been generated (Step III). On the other hand, during district generation, the algorithm does not annex a unit that has been excluded from a unit in the current solution (Step II). By passing over that unit, the algorithm does not examine as many possible combinations of population units, or branches on the tree, and, as a result, the execution time is faster. In general, for all the samples run, more population unit exclusions translate to faster execution time.

4.3 NEW MEXICO

Since the number of districts generated in the first phase of the algorithm tends to increase exponentially with an increase in population units, New Mexico, consisting of 33 counties, presents a higher level of difficulty than Nevada. New Mexico requires three districts and the mean district population is 434,298. The current plan, shown in Figure 4-5, has a maximum population deviation of 0.45% in the second district and a maximum shape compactness measure of 2.5, also in the second district.

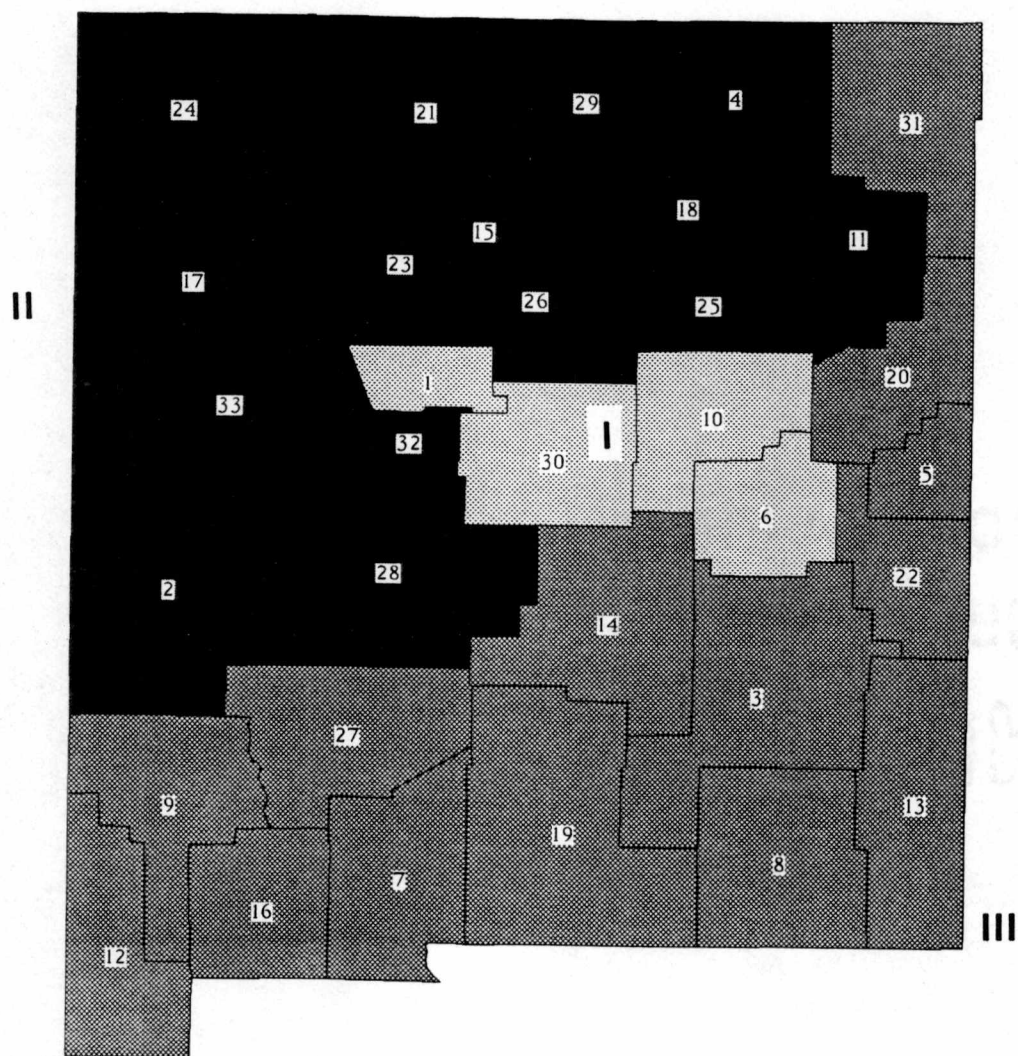


Figure 4-5: New Mexico current plan.

With the exception of population unit 1 (Bernallilo county), the population of New Mexico is distributed relatively evenly among the counties. Bernallilo county has a population of 419,700, which, if it were the only unit in a district, gives a 3.4% deviation from the mean district population. As a result of this concentration of population, there are relatively few "good" districts that contain Bernallilo county. On the other hand, the even population distribution throughout the remainder of the state results in a very large number of feasible districts containing counties other than Bernallilo county. To keep the solution process manageable within memory available on the VAX, the system is allowed to generate a maximum of 5000 feasible districts. Shown in Table 4-2 are all the computational results achieved for New Mexico during the runs described below.

The existence of the inordinately large number of feasible districts became apparent during an initial trial run in which the only constraining attribute was population deviation (and contiguity, of course). The maximum allowable population deviation (α) was set to 0.5% with no exclusion distance, no limit on shape compactness, and no enclave checking. Within 2 minutes of execution time, the algorithm had found over 5000 feasible districts, of which at least 481 had a population exactly equal to the mean population, that is, 0.00% population deviation! At this point, the run violated the maximum of 5000 districts and terminated. Since the algorithm had not yet even completed the generation of all feasible districts containing population unit 2, no attempt was made to find a solution in the second phase of the algorithm.

In an effort to compare the results produced by this system with the current plan, for the first run the parameters are set to allow the existing plan to be found by the system. Therefore, the maximum allowable population deviation is set to 0.45% while the maximum allowable measure of shape compactness (β) is set to 2.5. Since the two units in the state with the greatest distance between them are in the

Table 4-2: Computational Results for New Mexico.

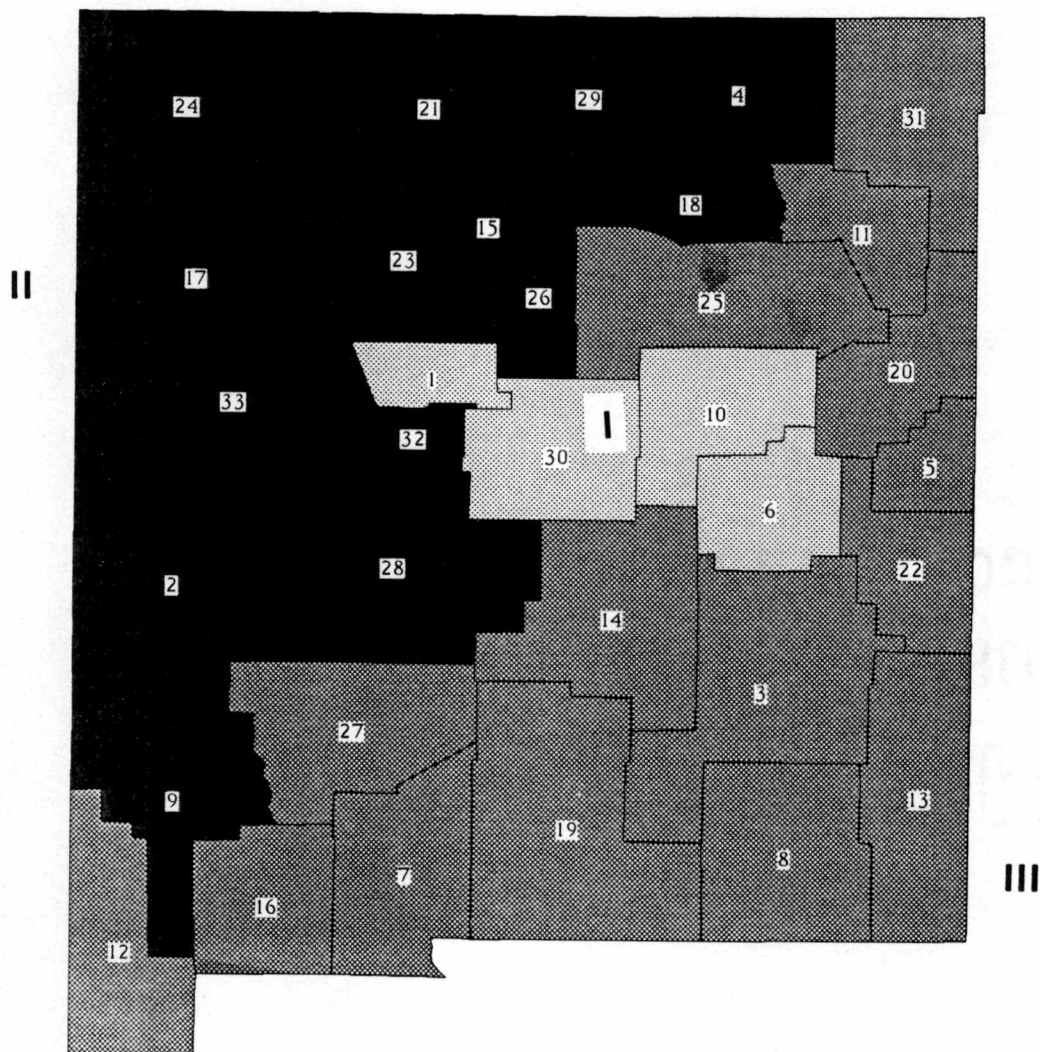
	Run 1 (Figure 4-6)	Run 2 (Figure 4-7)	Run 3 (Figure 4-8)
<u>INPUT</u>			
α	0.0045	0.0045	0.0045
β	2.5	2.5	2.5
EN	5	5	5
N1	3	3	3
Exclusion distance	∞	∞	300 miles
Boundary Exclusions	None	Figure 4-7	Figure 4-8
Boundary Inclusions	None	None	None
<u>OUTPUT</u>			
Districts generated	2112	1996	801
Phase I time	21.2 s	21.2 s	12.3 s
Phase II time	0.30 s	0.29 s	0.20 s
Maximum population deviation	0.0013(j=2)	0.0042(j=2)	0.00446(j=3)
Maximum shape compactness	2.3 (j=3)	2.2 (j=3)	1.7 (j=3)
Alternate optima	None	None	None

same district in the existing plan, no exclusion distances are used. To further limit the number of districts generated, enclave checks are performed every 3 iterations.

With these parameters, the algorithm found a plan, shown in Figure 4-6, that actually improves on both the population deviation and the shape compactness of the current plan. The plan has a maximum population deviation of 0.13% and shape compactness measure of 2.3. Even with the other parameters tightened, we note that the lack of exclusions, either distance or grouping, allows the algorithm to generate a large number of feasible districts during the first phase.

However, we note the unusual arrangement of counties in the southwestern corner of the state. In an arrangement that does not seem especially desirable, unit 12 (Hidalgo county) is not in the same district as unit 9 (Grant county). This arrangement provides opportunity for a practical application of boundary specification. For the second run, all parameters keep the same values as in the first run while the boundary between Hidalgo and Grant counties is selected to be excluded as a district boundary. In other words, Hidalgo and Grant counties must now exist in the same district in any plan found by the system. The plan found by the system during this run is displayed in Figure 4-7. Again, the system found a plan better than the existing plan in terms of both population deviation and shape compactness. The plan has a maximum population deviation of 0.42% and a maximum measure of shape compactness of 2.2. We note that only a relatively few districts (116) that were feasible in the first run become infeasible during the second run.

In addition to the parameters and boundary specification used in the second run, an exclusion distance of 300 miles is added to the constraints for the third run, for which the results are shown in Figure 4-8. As expected, the number of feasible districts generated and the execution time for the first phase decreases from the previous runs. Surprisingly, the plan found in this run is also an



Population Unit Grouping

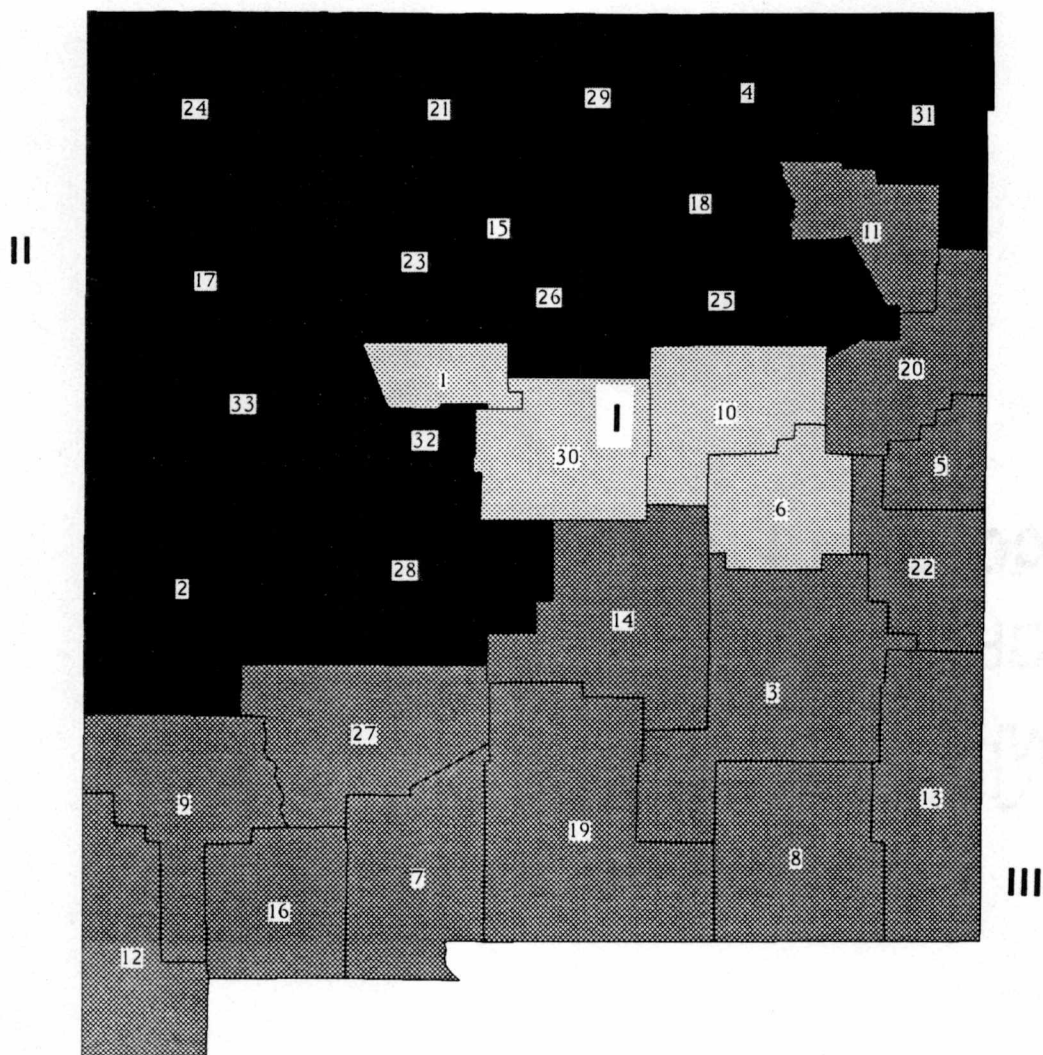
Boundary Exclusions

None.

Boundary Inclusions

None.

Figure 4-6: New Mexico - Run 1 results.



Population Unit Grouping

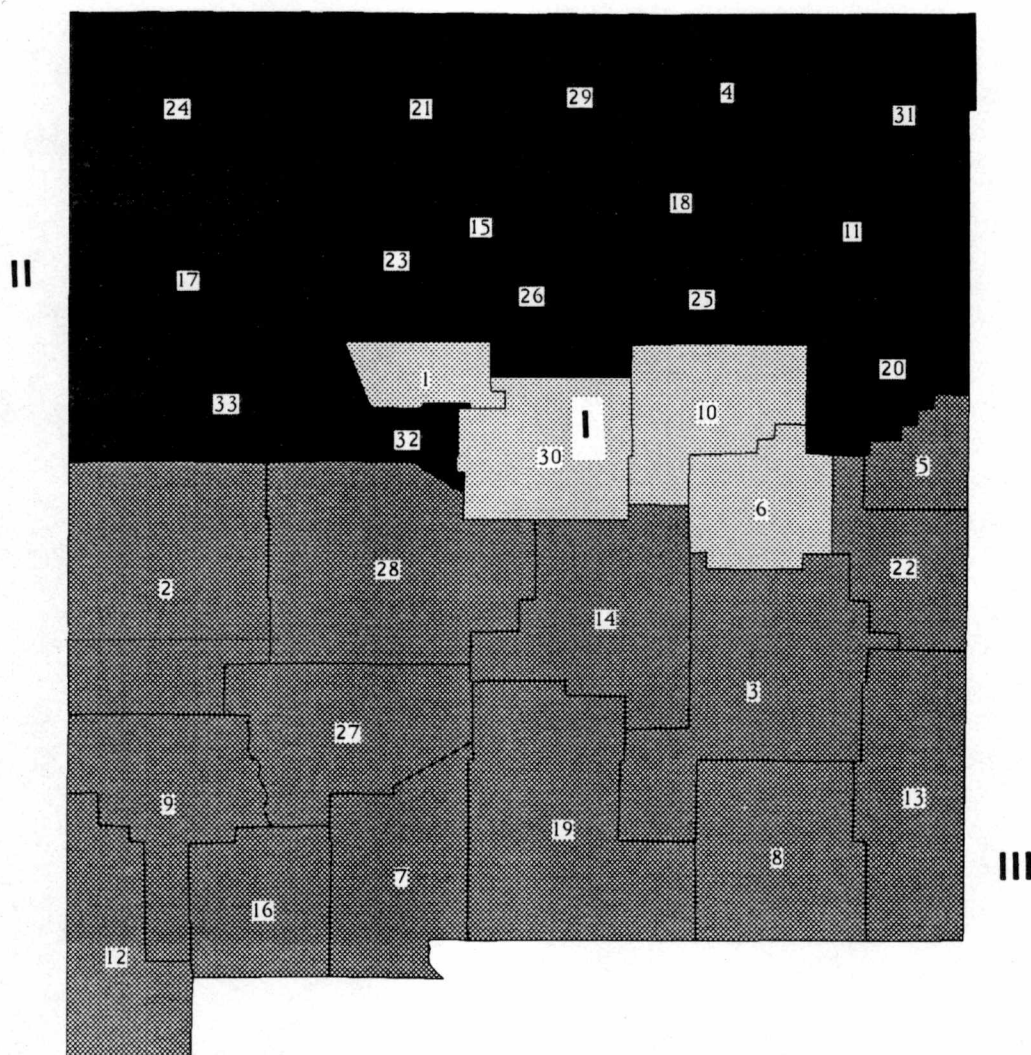
Boundary Exclusions

9 & 12

Boundary Inclusions

None.

Figure 4-7: New Mexico - Run 2 results.



Population Unit Grouping

Boundary Exclusions

9 & 12

Boundary Inclusions

None.

Figure 4-8: New Mexico - Run 3 results.

improvement over the existing plan, both in population deviation and shape compactness. This plan has a maximum population deviation of 0.44% and a maximum measure of shape compactness of 1.7.

4.4 WEST VIRGINIA

West Virginia consists of 55 counties and requires four districts. The mean district population is 487,411. Its current plan, shown in Figure 4-9, has a maximum population deviation of 0.27% in the third district and a maximum compactness measure of 4.3 in the second district.

The geography of West Virginia provides an excellent opportunity for utilization of boundary specifications in solving for a plan. The "arms" to the north and northeast are one county wide at the ends and therefore, the grouping of the seven counties in these arms is severely limited. Since the combined population of Berkeley, Jefferson, and Morgan counties (units 2, 19, and 33) is not enough to form a feasible district, they must exist in the same district as Hampshire county (unit 14). Similarly, Brooke, Hancock, Marshall, and Ohio counties (units 5, 15, 26, and 35) must exist in the same district as Wetzel county (unit 52). Because of these requirements, the boundaries between all these units are removed from consideration as district boundaries for all runs, thereby reducing the number of feasible districts found by the first phase. This reduction is especially important because of the large number of population units in West Virginia. Shown in Table 4-3 are all the computational results achieved for West Virginia during the runs described below.

The first run uses parameters comparable to the characteristics for the current plan in an attempt to compare any solution found by the system with the existing plan. The maximum allowable population deviation (α) is set to 0.30% and the maximum allowable measure of shape compactness (β) is set to 4.5.

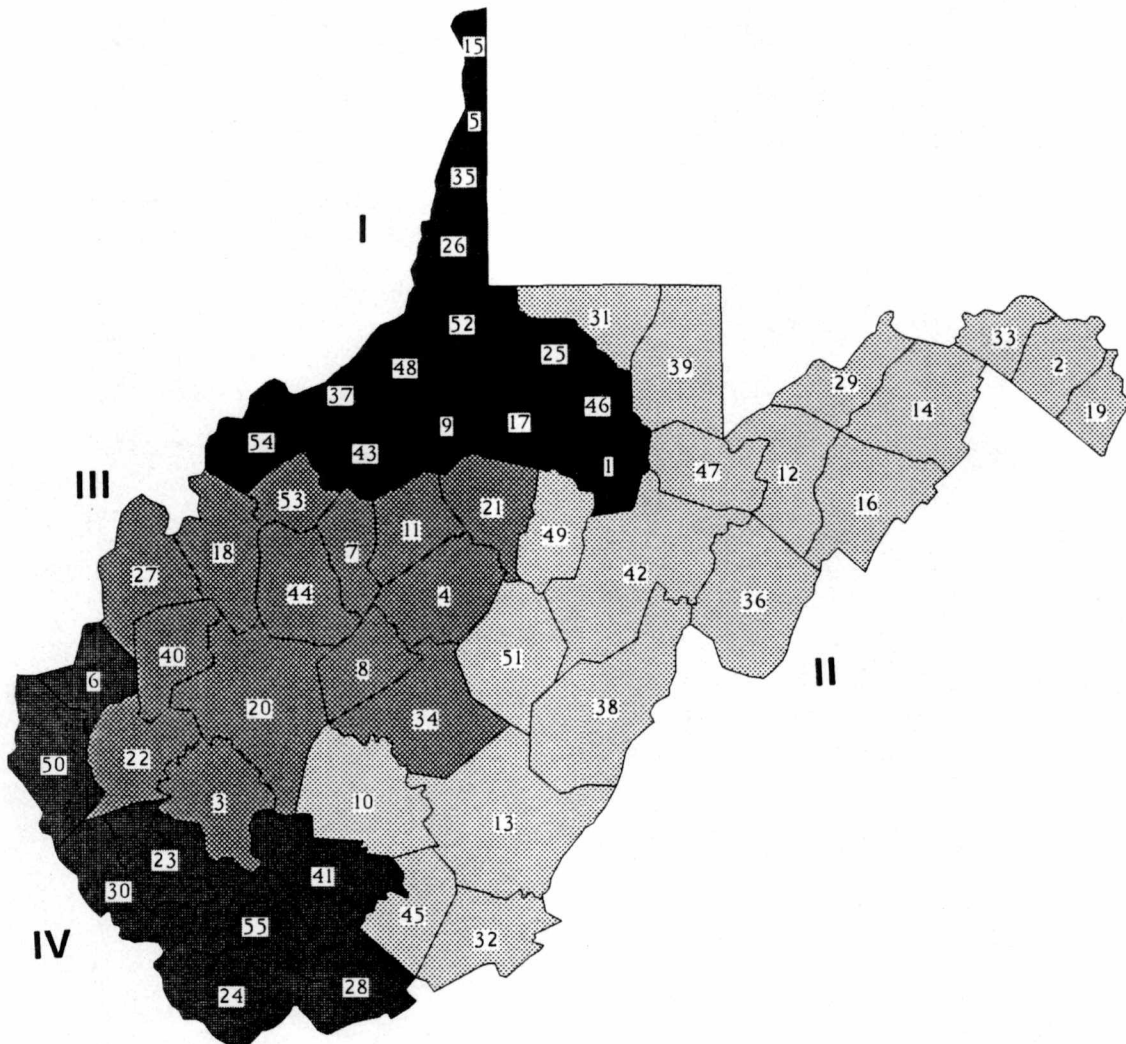


Figure 4-9: West Virginia current plan.

Table 4-3: Computational Results for West Virginia.

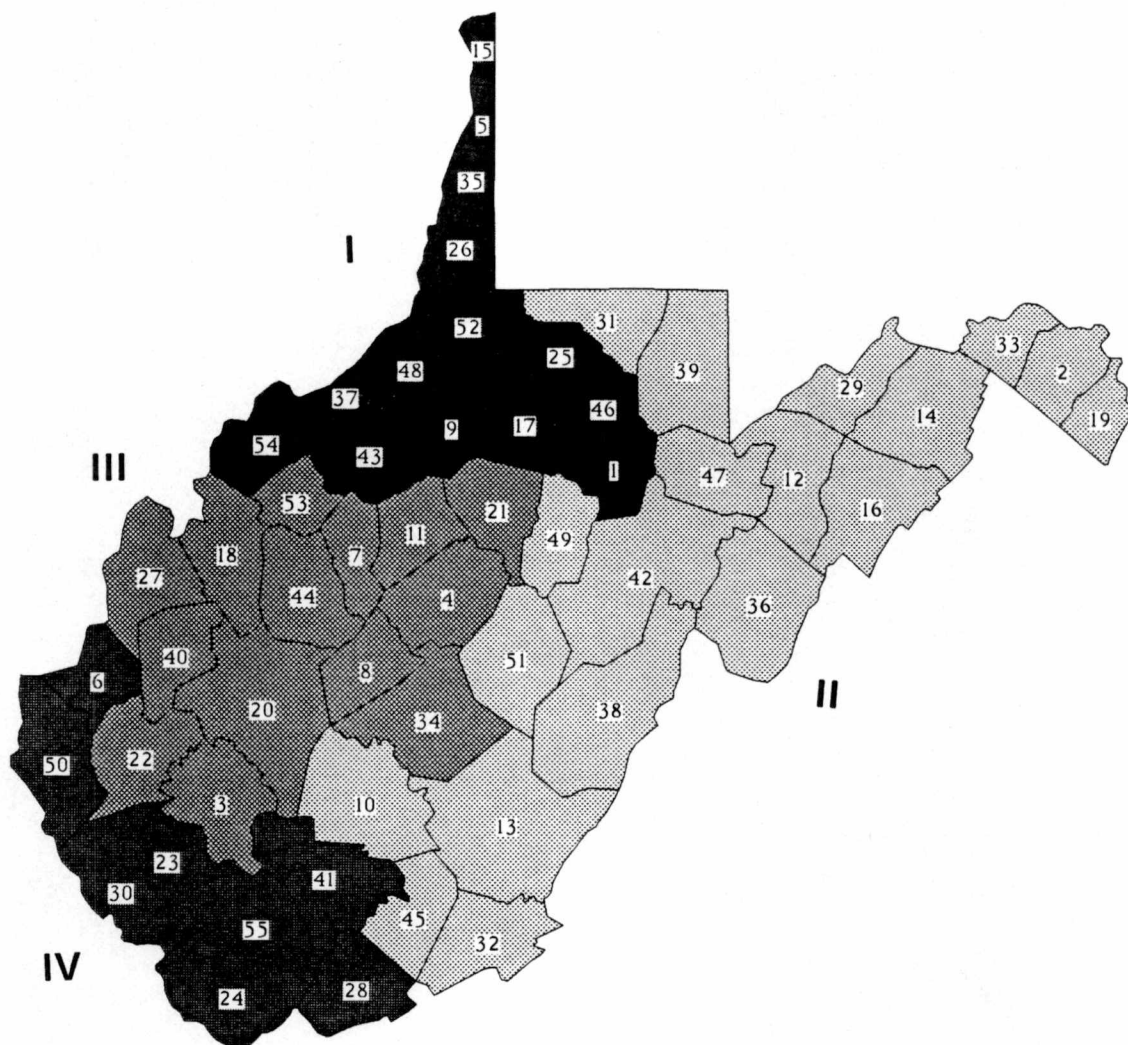
	Run 1 (Figure 4-10)	Run 2 (Figure 4-11)	Run 3 (Figure 4-12)
<u>INPUT</u>			
α	0.0030	0.0020	0.0020
β	4.5	5.0	7.0
EN	5	5	5
N1	3	3	3
Exclusion distance	200 miles	200 miles	220 miles
Boundary Exclusions	Figure 4-10	Figure 4-11	Figure 4-12
Boundary Inclusions	None	None	Figure 4-12
<u>OUTPUT</u>			
Districts generated	3589	3055	1459
Phase I time	7.9 min	6.8 min	3.7 min
Phase II time	5.25 s	4.82 s	3.04 s
Maximum population deviation	0.0027(j=3)	0.0015(j=3)	0.0006(j=4)
Maximum shape compactness	4.3 (j=2)	5.0 (j=2)	6.6 (j=1)
Alternate optima	None	None	None

Additionally, an exclusion distance of 200 miles is specified. As shown in Figure 4-10, the plan found in the first run exactly matches the existing plan. In other words, the current districting plan for West Virginia is optimal with respect to population deviation and shape compactness. We note the numerous feasible districts generated in the first phase even with the tight restrictions on population deviation and shape compactness.

The parameters are set for a second run to allow the system to search for a plan that improves on the population deviation of the current plan. To that end, the maximum allowable population deviation is set to 0.2%. Since the first run produced no plans better than the existing plan, β is increased to 5.0. The plan found by the algorithm in the second run has a maximum population deviation of only 0.15% but the measure of shape compactness increases to 5.0. Figure 4-11 displays the plan found by the system for this run. Again, we note the large number of feasible districts generated in the first phase despite the tight restrictions.

Even though the plan found is an improvement over the existing plan in terms of population deviation, the composition of the second district in the plan does not seem desirable. The population of the second district is distributed among counties throughout the state, including counties on the south, east, and northwest borders of the state.

The plan shown in Figure 4-12 is the result of an attempt to improve the geography of the plan and examine the impact of more boundary specifications on the solution process. To accomplish these objectives, the second district found during the second run is "snipped" between units 9 and 21. That is, the boundary between units 9 and 21 is selected to be included as a district boundary and so units 9 and 21 may not exist in the same district. In addition to this boundary specification, units on either side of the Allegheny Mountains in the southeast



Population Unit Grouping

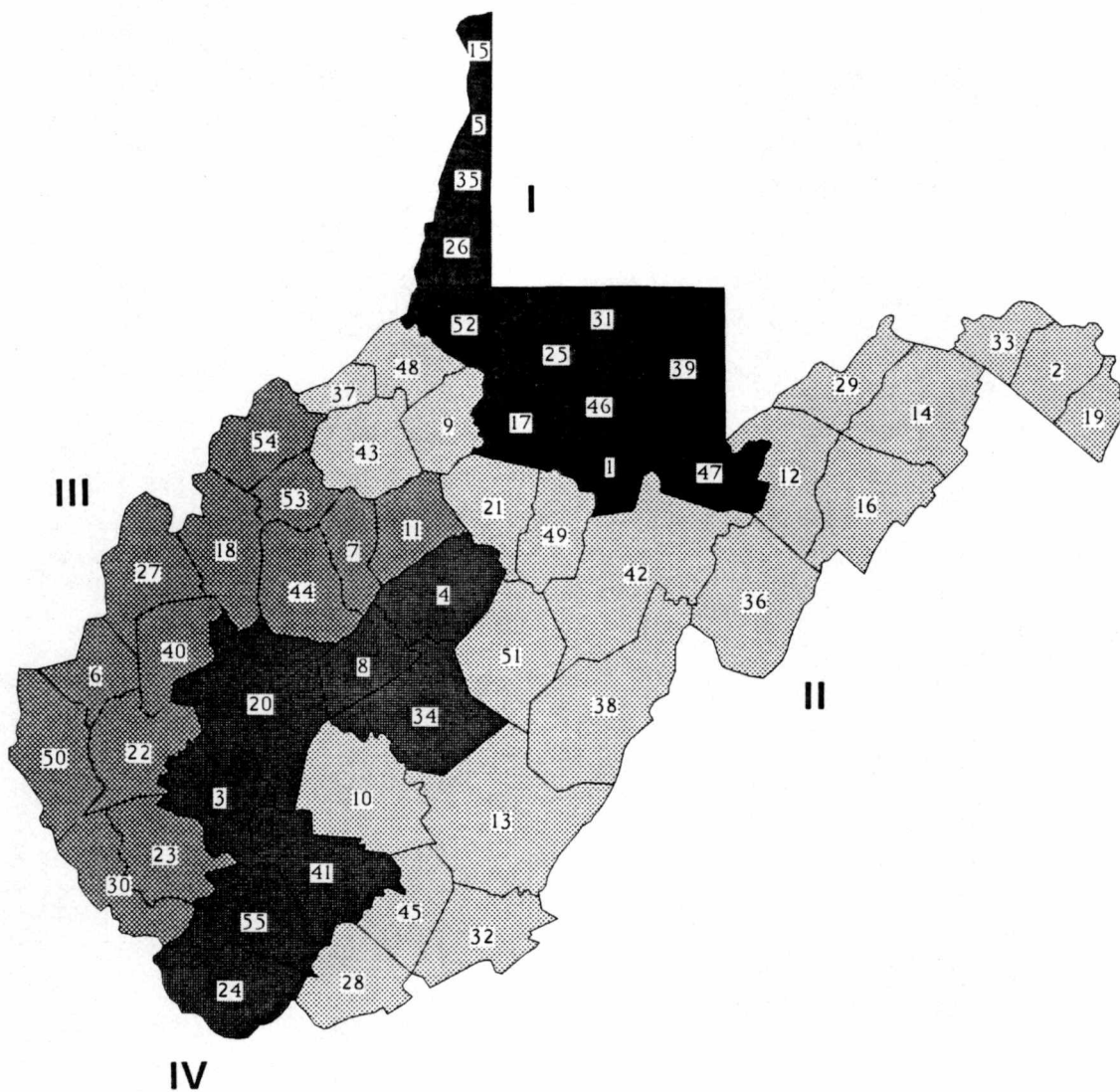
Boundary Exclusions

5 & 15, 5 & 35, 35 & 26, 26 & 52
2 & 19, 2 & 33, 14 & 33

Boundary Inclusions

None.

Figure 4-10: West Virginia - Run 1 results.



Population Unit Grouping

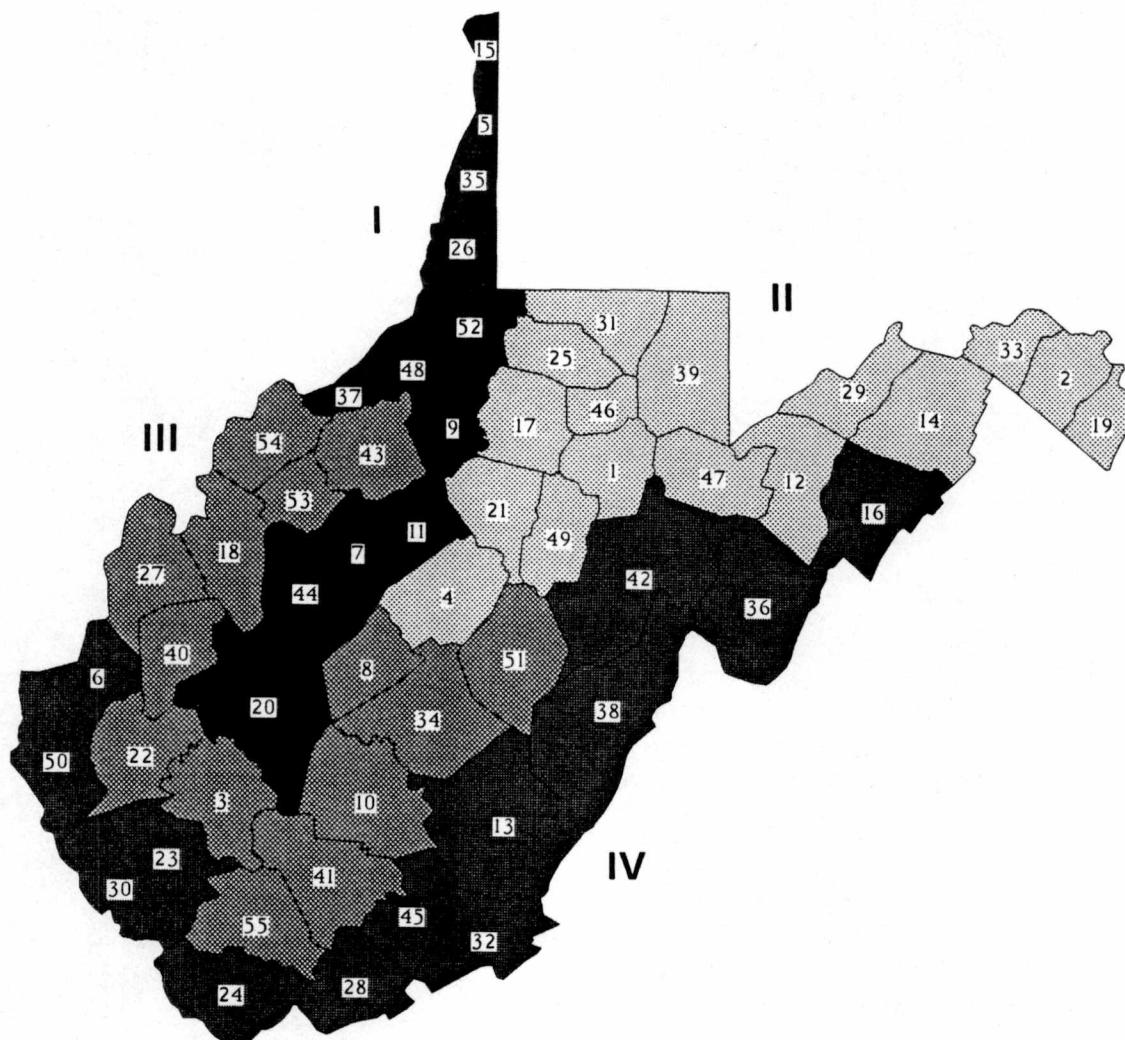
Boundary Exclusions

5 & 15, 5 & 35, 35 & 26, 26 & 52
2 & 19, 2 & 33, 14 & 33

Boundary Inclusions

None.

Figure 4-11: West Virginia - Run 2 results.



Population Unit Grouping

Boundary Exclusions

5 & 15, 5 & 35, 35 & 26, 26 & 52
2 & 19, 2 & 33, 14 & 33

Boundary Inclusions

9 & 21, 42 & 51, 38 & 51
10 & 13, 13 & 34, 41 & 45

Figure 4-12: West Virginia - Run 3 results.

corner of the state are prohibited from existing in the same district. The boundaries between the units on either side are also selected to be included as district boundaries. The maximum population deviation remains at 0.20%. However, in anticipation of worsening geography for the plan, the exclusion distance is increased to 220 miles and β is initially increased to 6.0. With this value for β , no plan was found so β is increased to 7.0.

Because of the exclusions specified during this run, we note the decrease in the number of feasible districts generated and the reduction in execution time in the first phase. Having a maximum measure of shape compactness of 6.6, the plan found is very poor with respect to shape compactness. Interestingly, even with the restrictions on population unit grouping, this plan is the best of any plan found in terms of population deviation. The plan has a maximum population deviation of only 0.06% ! While the geography of this plan makes is undesirable, it does demonstrate two recurring effects, the trade-off between population deviation and compactness and the reduced solution time because of the number of specified exclusions.

The state of West Virginia was also used as a test problem for the original implementation. The results obtained here contrast the current system with the original implementation in that the original implementation was not able to solve the 55 population unit problem. In the original work, two approaches were utilized in an effort to solve the problem. First, a heuristic solution was found by hand to generate reasonable exclusion distances. The Phase I run using these distances was not nearly complete in 30 minutes of execution time and this attempt was abandoned. The second approach allowed the exclusion distances to vary over the state. Runs using this approach either became computationally infeasible or a unit became uncovered as a result of too restrictive exclusions. The results obtained by the current implementation do confirm, as Garfinkel stated, "that with sufficient

effort and computer funds a feasible set of districts" may be found. (Garfinkel, 1968, p.77) Specifically, two plans were found with the current implementation using only boundary specifications that must occur in any plan due to the geography of the state. While the results of the third run utilizing additional boundary specifications may be sub-optimal to the original problem, it is clear that the plan is still very good with respect to population deviation.

4.5 WASHINGTON

Washington has 39 counties and requires 8 districts. Because of the large populations of King County and Pierce County (units 17 and 27), its current plan does not maintain county boundaries. A representation of the current plan is given in Figure 4-13. The existing plan has a maximum population deviation of 0.45%. The population is 4,132,156, resulting in a mean district population of 515,394.5. Since King County has a population of 1,269,749, a single member, eight district plan maintaining county boundaries is clearly not feasible. Two approaches to determination of a plan for Washington are considered.

In general, the first approach first solves the problem as a 6 district problem and then solves a subproblem as a 3 district problem. There are two definitions of the problem using this approach.

The first definition involves subtracting twice the population mean ($2\bar{p}$) from King County (unit 17), leaving it with a population of 216,710 and solving the problem as a 6 district problem. Then $2\bar{p}$ is added back to the resulting district containing King County and this subproblem is solved as a 3 district problem using smaller population units, such as census tracts, enumeration districts, or county divisions.

The second definition combines the population of King county with the population of a contiguous county and districts the two counties into three

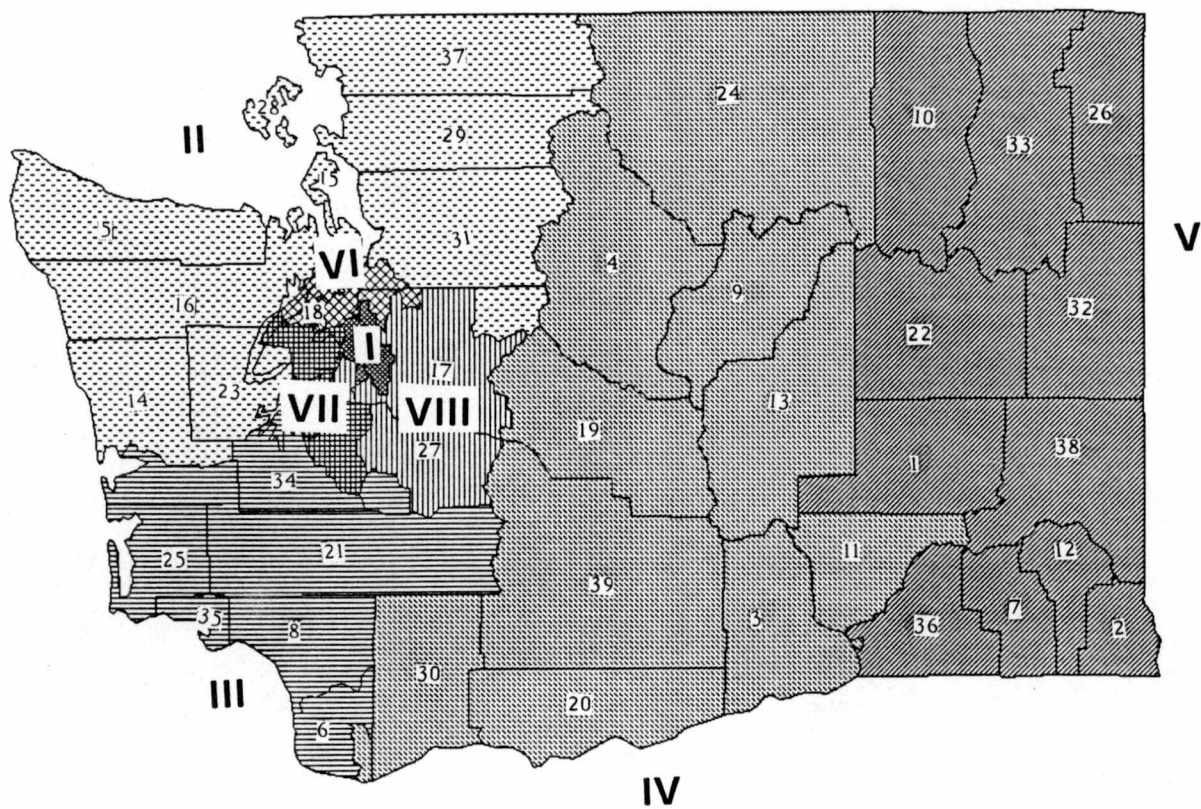


Figure 4-13: Washington current plan.

districts. However, addition of Pierce County (unit 27) results in a population of 1,755,392. If we make three districts out of these two counties, the mean population for the three would be 585,130, or a 13.2% population deviation. Creating four districts results in a 17.3% population deviation. Similar checks were made for each surrounding county, with no combination providing less than 13.2% population deviation. Since the existing plan has a deviation of only 0.45%, an approach assuring a deviation of at least 13.2% does not seem desirable.

The second approach is to utilize the county subdivisions that are designated by the Bureau of the Census as County Divisions. For solution, each county, or at least the most populous ones, are subdivided and these smaller units are used as the population units. King County consists of ten County Divisions. However, the Seattle division has a population of 744,792 which is still greater than the mean district population and so a further breakdown into Census Designated Places (CDPs) is required. Using the 27 CDPs for the Seattle division in addition to the other 9 County Divisions as population units increases the number from 39 units to 74 population units. Using these subdivisions as population units, no successful runs were completed within the available computer resources. While this approach seems desirable, it unfortunately results in a number of population units that is beyond the current capability of the system. While it is not clear exactly what the limits of the system are, we know the system solves the 55 population unit problem of West Virginia but cannot solve the 74 population unit problem as described above.

As a result, the first approach discussed is the approach utilized in the following runs, for which the computational results are listed in Table 4-4. Since digitized boundary points for the county subdivisions are not readily available, all resulting maps for Washington display the 6 district plan found by the system for each run.

Table 4-4: Computational Results for Washington.

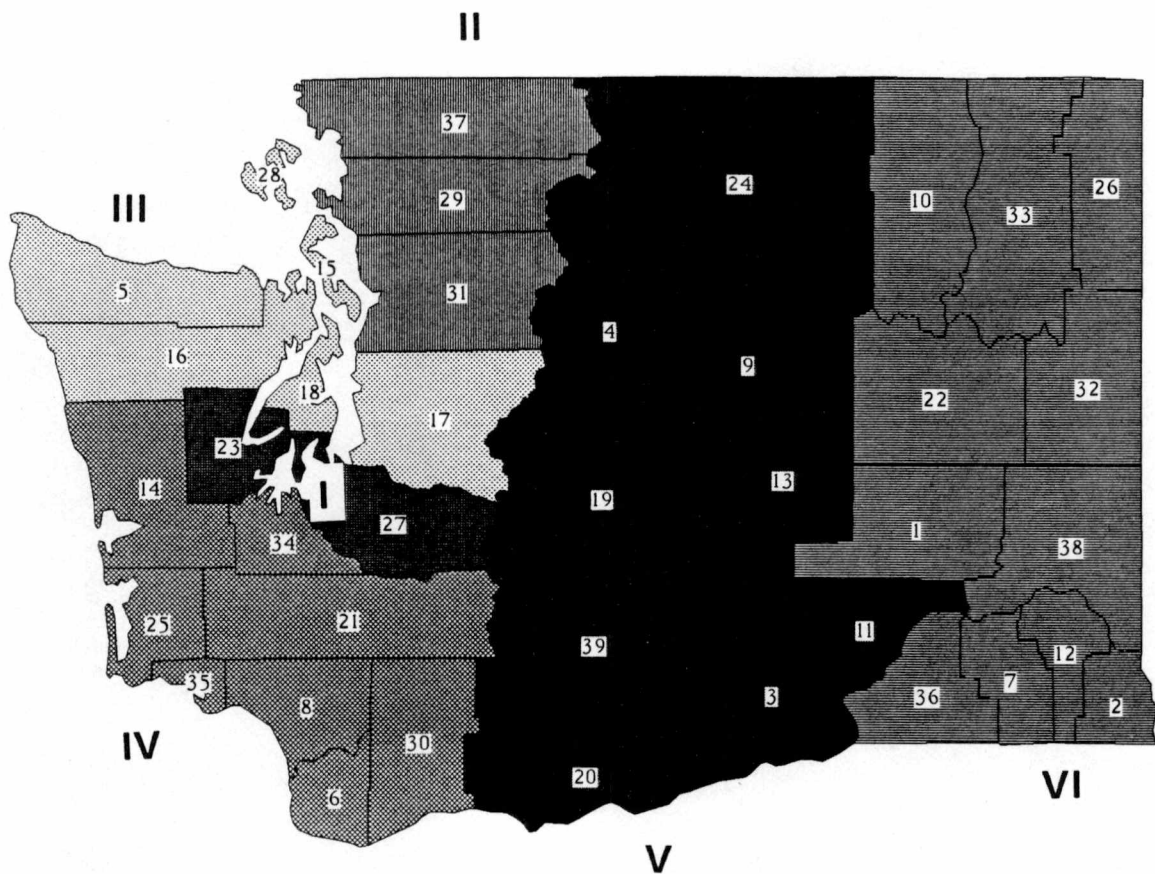
	Run 1 (Figure 4-14)	Run 2 (Figure 4-15)	Run 3 (Figure 4-16)
<u>INPUT</u>			
α	0.010	0.020	0.040
β	∞	∞	1.5
EN	3	3	3
N1	5	5	5
Exclusion distance	250 miles	250 miles	250 miles
Boundary Exclusions	None	None	None
Boundary Inclusions	None	Figure 4-15	Figure 4-16
<u>OUTPUT</u>			
Districts generated	1522	806	232
Phase I time	2.2 min	1.5 min	58.3 s
Phase II time	0.30 s	0.28 s	0.22 s
Maximum population deviation	0.0070(j=3)	0.0181(j=4)	0.0362(j=1)
Maximum shape compactness	3.5 (j=5)	1.9 (j=6)	1.5 (j=4)
Alternate optima	None	None	None

Because of the solution approach, no direct comparisons with the existing plan are possible. Also, in initial runs, it became apparent that the existence of several populous units in close proximity, such as King and the surrounding counties, limits the number of "good" districts with respect to population deviation. Shown in Figure 4-14 is the plan found in the first run, for which the maximum allowable population deviation is set to 1.0% with a maximum allowable measure of shape compactness of infinity. Additionally, an exclusion distance of 250 miles is used and enclave checks are performed every 5 iterations.

The plan found during the first run has a maximum population deviation of 0.7% and a maximum measure of shape compactness of 3.5. We note the increased population deviation found by system as compared to the existing plan's maximum population deviation is 0.45%. While the plan found is still a "good" one, having less than 1.0% population deviation, all plans found by the system using our 6 district approach are worse than the existing plan with respect to population deviation. This fact indicates the need for further improvements to the system in terms of the number of population units the system can handle.

We note the second district of the plan found in the first run includes units 17 and 19, which are on either side of the Cascade mountains. As a practical application of boundary specifications and to further demonstrate the reduction in execution time for the first phase when pairs of population units are excluded from each other, the second run specifies that units on either side of the Cascade Mountains cannot exist in the same district. In other words, the boundaries between counties that border on the mountains are selected to be included as district boundaries. These boundary specifications as well as the results for this run are shown in Figure 4-15.

Since no restriction on the shape compactness measure is specified for either run, we expect the maximum population deviation for this plan to be worse than



Population Unit Grouping

Boundary Exclusions

None.

Boundary Inclusions

24 & 37, 24 & 29, 4 & 31, 4 & 17, 17 & 19
27 & 39, 21 & 39, 30 & 39, 20 & 30

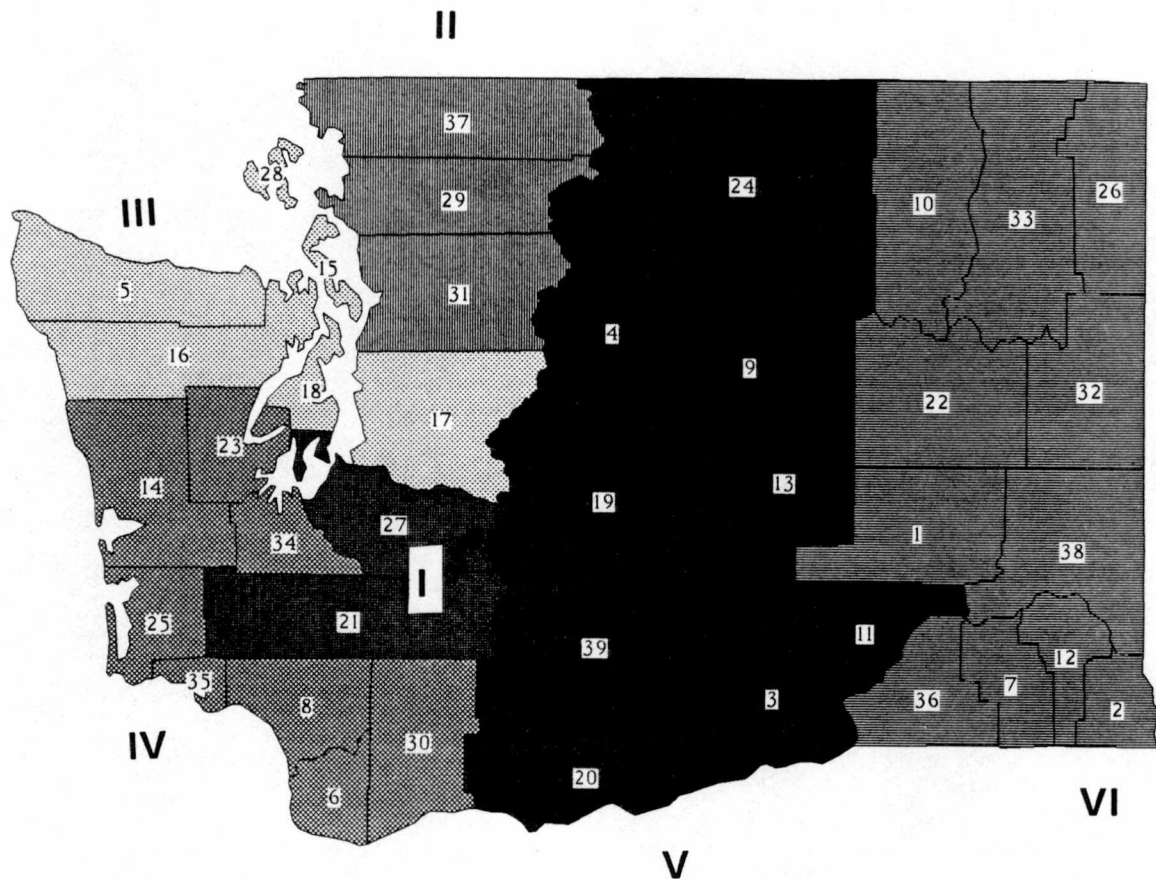
Figure 4-15: Washington - Run 2 results.

that for the plan found in the first run. Indeed, the maximum population deviation increases to 1.8%. Interestingly, the geography of the state combined with the boundary specifications result in a sizable improvement in the maximum measure of shape compactness to 1.9.

Finally, in a geographical exercise, an attempt is made to find the minimal shape compactness for any plan found by the system. To achieve this goal, the maximum measure of shape compactness is reduced to 1.5. In anticipation of the trade-off between compactness and population deviation, the maximum allowable population deviation is increased to 4.0%. All of the boundary specifications made during the second run are allowed to remain and an optimal plan is found by the system. This geographically compact plan is displayed in Figure 4-16. The plan found by the system lowered the maximum measure of shape compactness to 1.5 while the population deviation increases to 3.6%. Again, we note the trade-off between shape compactness and population deviation.

As another demonstration of the effects of boundary exclusions, another run for these parameters was made, removing the boundary specifications along the Cascade Mountains that were previously included. In other words, any combination of population units that is feasible with respect to population and compactness may exist as a district. Demonstrating the increased solution time and solution size without exclusions, the first phase of this run took longer (2.0 minutes) and generated more feasible districts (664) than the previous run, arriving at the same solution.

As with West Virginia, the state of Washington was also used as a test problem for the original implementation. The original work, based on the 1960 Census data, solved for 7 districts rather than the current 8 districts. As with the current problem, a single-member districting plan maintaining counties as population units was infeasible due to the population of King County. The



Population Unit Grouping

Boundary Exclusions

None.

Boundary Inclusions

24 & 37, 24 & 29, 4 & 31, 4 & 17, 17 & 19
27 & 39, 21 & 39, 30 & 39, 20 & 30

Figure 4-16: Washington - Run 3 results.

solution approach used in the original work is essentially the same approach that is used in the current work. A contiguous county (Pierce County) was added to King County and three districts were made out of these two counties. Then the remaining 37 county problem was solved to obtain 4 districts. (Garfinkel, 1968)

The contrast between the original results and the current results is not as dramatic as in the case of West Virginia. In other words, the original implementation was able to successfully solve for a plan for Washington. Despite the changes in population and its distribution from 1960 to 1980, both implementations found a similar number of feasible districts during runs in which population deviation and exclusion distances played the constraining roles. The original implementation found 1648 feasible districts compared to 15522 districts found by the current implementation. As expected, the current system's execution time of 2.2 minutes is better than the original system's execution time of 3.1 minutes. The use of boundary specifications in the current system allows subsequent solutions to be found easier and quicker than in the original system. However, with more efficient data structures and source code, the differences in solution time could and should be several magnitudes greater than the results obtained during these runs.

A final difference between the results of the two systems occurs during runs attempting to find the plan with the minimum measure of shape compactness. While Phase I of the original system terminated after 9 minutes of execution time when a unit became uncovered, the current system, using boundary specifications, solves the first phase in 58.3 seconds. The changes in the distribution of the state's population and the different number of required districts are the probable reasons that the original system was unable to arrive a solution while the current system did find a plan. The substantial difference in execution time is primarily due to the use of boundary specifications in the current system.

4.6 CONCLUSIONS

The results obtained by the system during the experiments for Nevada, New Mexico, West Virginia, and Washington indicate both promise and areas of improvement for this system. The use of boundary inclusions and exclusions afford the ability to break larger problems into smaller ones. However, as was the case in all four states, too many boundary specifications result in the system finding no plan.

This algorithm and implementation enhances the original work by solving larger problems and reducing execution time for solution. The original implementation did not find a solution for West Virginia, whereas this implementation did find plans for the state. Even though these plans may be sub-optimal to the original problem, the system is able to achieve very good (or possibly optimal) solutions to problems larger than the original implementation could handle. However, there is still a limit to the size of the problem the system can solve. For example, the system could not solve the 74 population unit problem for Washington subdivided into county divisions. Additionally, even though the execution times were faster than the original implementation, much of this increase is probably due to the faster and more powerful computer. Including more efficient data structures and some type of assembly language coding or bit manipulation to the current system should further reduce the execution times.

Another enhancement to the system would reduce the number of feasible districts generated and reduce the execution time for the first phase, at least somewhat. That enhancement would consist of using the current existing plan or implementing a "greedy" heuristic to preprocess the state's data and find a good (but not necessarily optimal) starting plan. The value for this plan would serve as an upper bound for the districts found in the execution of the first phase of the

algorithm. In other words, any district generated with a value greater than the value for the starting plan would be discarded, since at that point the algorithm knows there is a better solution. Setting such an upper bound in combination with the population unit grouping constraints described in this paper could be utilized to greatly reduce the number of districts generated in the first phase.

Additional potential improvements to the system include improved graphics, the use of some type of database management system and implementation of the system, or a similar one on a personal computer.

The states investigated have already achieved good, or even optimal, districting plans, by whatever means. The elements of this system, particularly the interactive graphics, may be beneficial in determining other desirable districting plans.

LIST OF REFERENCES

- Baker, Gordon E. 1973. Quantitative and Descriptive Guidelines to Minimize Gerrymandering. *Annals, New York Academy of Sciences* 219, 200-208.
- Bodin, Lawrence D. 1973. A Districting Experiment with a Clustering Algorithm. *Annals, New York Academy of Sciences* 219, 209-214.
- Chance, C. 1965. *Political Studies: Representation and Reapportionment*. Columbus: Ohio State University Press.
- Forrest, E. 1964. Apportionment by Computer. *American Behavioral Science* 7, 52-54.
- Garfinkel, Robert S. and G.L. Nemhauser. 1970. Optimal Political Districting by Implicit Enumeration Techniques. *Management Science* 16, B495-B507.
- Garfinkel, Robert S. 1968. *Optimal Political Districting*. Ph.D Thesis, The Johns Hopkins University.
- Gearhart, Burton C. and John M Liittschwager. 1969. Legislative Districting by Computer. *Behavioral Science* 14, 404-417.
- Goldwin, Robert A., (ed). 1968. *Representation and Misrepresentation*. Chicago: Rand McNally and Company.
- Grofman, Bernard, Arend Lijphart, Robert B. McKay, and Howard A. Scarrow, (ed). 1982. *Representation and Redistricting Issues*. Lexington, MA: Lexington Books.
- Hess, S.W. and J.B. Weaver, H.J. Siegfeldt, J.N. Whelan, and P.A. Zitlau. 1965. Nonpartisan Political Redistricting by Computer. *Operations Research* 13, 998-1006.
- House, William C. (ed). 1982. *Interactive Computer Graphics Systems*. New York: Petrocelli Books.
- Liittschwager, John M. 1973. The Iowa Redistricting System. *Annals, New York Academy of Sciences* 219, 221-235.

- Morrill, Richard L. 1981. *Political Redistricting and Geographic Theory*. Washington, D.C.: Association of American Geographers.
- Nagel, S.S. 1965. Simplified Bipartisan Computer Redistricting. *Stanford Law Review* 12, 863-899.
- Savas, E. S. 1964. A Computer-based System for Forming Efficient Election Districts. *Operations Research* 12, 135-155.
- Thoreson, J., and John M. Liittschwager. 1967. Legislative Districting by Computer Simulation. *Behavioral Science* 12, 237-247.
- Tobler, W. R. 1973. A Continuous Transformation Useful for Districting. *Annals, New York Academy of Sciences* 219, 215-220.

VITA

Loren Scott Buckels grew up in Gautier, Mississippi. After graduating from Pascagoula High School in 1980, he attended Tulane University in New Orleans, Louisiana. In May, 1984 he received his Bachelor of Science in Biomedical Engineering from the School of Engineering at Tulane University.

After working for a year as a chemical researcher, he moved to Knoxville, Tennessee for his graduate studies at the University of Tennessee. He is currently employed at the Oak Ridge National Laboratory with the Transportation and Operations Research Planning Group in the Energy Division.