# Geographic Optimization Using Evolutionary Algorithms

Ningchuan Xiao

Department of Geography, The Ohio State University, Columbus, OH 43210, Email: xiao.37@osu.edu

**Abstract**

During the last two decades, evolutionary algorithms (EAs) have been applied to a wide range of optimization and decision-making problems. Work on EAs for geographic analysis, however, has been conducted in a problem-specific manner, which prevents an EA designed for one type of problem to be used on others. The purpose of this paper is to describe a framework that unifies the design and implementation of EAs for different types of geographic optimization problems. The key element in this framework is a graph representation that can be used to formally define the spatial structure of a broad range of geographic problems. Based on this representation, spatial constraints (e.g., contiguity and adjacency) of optimization problems can be effectively maintained, and general principles of designing evolutionary algorithms for geographic optimization are identified. The framework is applied to an example political redistricting problem.

## 1 Introduction

Geographers have placed a significant amount of attention on the development of methods that are applied to optimization problems, especially those that require search for optimized configurations of spatial entities and activities. These problems include environmental policy making (Bennett et al. 2004), locational analysis (Scott 1970; Rushton 1988; Densham and Rushton 1996), nature reserve selection (Church et al. 1996), natural resource management (Hof and Bevers 1998), redistricting (Williams 1995), spatial data mining and exploratory analysis (Han et al. 2001), spatial decision making in public and private sectors (Armstrong et al. 1991; Crossland et al. 1995; Ghosh and Harche 1993), and transportation (Miller and Shaw 2001). In this paper, these problems are loosely called geographic optimization problems. The search for solutions to such problems often involves intensive computation, and a variety of approaches have been developed to support their analysis.

Ideally, an approach to solving optimization problems should exhibit the following properties:

- Efficiency. Optimal solutions must be found in an efficient manner. More formally, the time required to find optimal solutions should be polynomial to the size of the problem.
- Optimality. Optimal solutions should always be found.

- Equality. Optimal solutions should be found for all instances of the problem.
- Near-optimality. Many geographic optimization problems contain important social, economic, and political factors that are difficult, if not impossible, to place in a mathematical formulation. Optimal solutions to mathematically well formulated problems will become non-optimal when the unmodeled factors or objectives are taken into account. Therefore, the near-optimal (or second best) solutions to the problem may become more favorable to the satisfaction of decision-makers (Simon 1960; Brill 1979; Hopkins 1984).

Unfortunately, no optimization algorithm has been found to satisfy all of the above requirements. In general, current solution approaches can be placed into three categories, exact, approximation, and heuristic, each of which is often designed to fulfill a subset of these requirements.

Exact algorithms have been employed to search for optimal solutions to geographic optimization problems (see, for example, Scott 1971; Daskin 1995). However, the time required to find an optimum solution using an exact algorithm often grows exponentially with problem size, and many real-world optimization problems cannot be solved in a reasonable period of time (Garey and Johnson 1979). In addition, many exact algorithms are generally not constructed to yield near-optimal solutions that are desired for many applications.

Approximation algorithms drop the optimality requirement — they do not require that optimal solutions always be found, though they guarantee that the best solution found is within a certain theoretical bound from the optimal solution for all instances of a problem (Vazirani 2001). These algorithms are usually based either on greedy approaches or on the relaxation of exact algorithms, notably linear programming. However, there are problems for which efficient approximation algorithms are difficult to devise and their theoretical bounds are difficult to prove.

The third type of algorithm, called heuristics, cannot guarantee the closeness to optimum for the solutions found. Nevertheless, they can be used to find a number of high quality (near-optimal) solutions, even though they may not always find the global optimal solution to a problem (Cooper 1964). Recent developments have been aiming to design general and flexible approaches to solving a broad range of optimization problems. These general methods, called metaheuristics (Osman and Kelly 1996) or modern heuristics (Reeves 1993), are often based on natural processes (e.g., biological evolution) and they typically include tabu search (Glover 1989; Glover 1990), simulated annealing (Kirkpatrick et al. 1983), ant colony systems (Dorigo et al. 1991), and evolutionary algorithms (Bäck et al. 1997).

Of these metaheuristic approaches, evolutionary algorithms have shown great promise for calculating solutions to large and difficult optimization problems and have been successfully used across a variety of problems in various applications (Goldberg 1989; Bäck et al. 1997). The use of evolutionary algorithms in geographic problem-solving has been investigated by a number of researchers during the last two decades (Hosage and Goodchild 1986; Dibble and Densham 1993; Bennett et al. 1999; Krzanowski and Raper 1999; Brookes 2001; Jaramillo et al. 2002; Xiao et al. 2002; Bennett et al. 2004). However, the evolutionary approaches described in the literature are based on representations and operations that are specific to each particular problem being addressed by the research.

The purpose of this paper is to develop a unified framework that can be used to guide the design and implementation of evolutionary algorithms for solving geographic optimization problems. This approach is based on a general framework that can be used to formalize the

representation of different problems. In the remainder of this paper, Section 2 discusses how this framework is used to formulate the optimization problems in geographic analysis. Section 3 briefly introduces the principles of evolutionary algorithms, while Section 4 describes how these algorithms can be used to solve geographic problems. In Section 5, the use of the framework is demonstrated using an example political redistricting problem. Finally, we conclude this paper by positioning it in the context of recent development in computational geography.

# 2 Formulating Geographic Optimization Problems

Two broad types of geographic optimization problems can be generally identified: those that require the partitioning (or grouping) of spatial entities and those that require the selection of a subset of spatial entities. For each type of problem, two subtypes can be defined according to whether the partitioning or selection is spatially constrained. One may argue that a selection problem is a special case of a partitioning problem, in which the spatial units are partitioned into two subdivisions (one formed by selected entities, and the other by unselected entities). However, it should be noted that when spatial constraints are required, they normally do not apply to the unselected units. Therefore, it is necessary to distinguish selection and partitioning problems. The four types are summarized below.

- Selection problems without spatial constraints. Among a number of spatial entities, a subset is selected to satisfy a set of objectives. There is no specific spatial requirement about how these selected entities should be organized in a space. Typical examples are the *p*-median problem (Hakimi 1965) and set-covering problem (Schilling et al. 1993).

- Selection problems with spatial constraints. Entities selected must comply with some spatial constraints. Site selection problems are an example where selected entities must be contiguous (Cova and Church 2000; Xiao et al. 2002). Other examples include the harvest problem in forestry management where selected entities (e.g., plots) must not be adjacent (Hof and Bevers 1998; Murray and Church 1995).

- Partitioning problems without spatial constraints. For this type of problem, each spatial entity is assigned a value and the goal is to find a combination of values for all entities so that a set of objectives can be optimized. A typical example is the study of optimal landscapes by Bennett et al. (2004).

- Partitioning problems with spatial constraints. For this type of problem, the space must be partitioned in a certain way such that some spatial constraints are satisfied. Typical examples include political redistricting problems where each district must be contiguous (Williams 1995; Altman 1998).

A graph can be used to formulate geographic optimization problems. Here, a graph is defined as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of $n$ vertices, $E$ is a set of edges with each edge comprised of two vertices, and edge $(v_i, v_j) \in E$ if and only if vertices $v_i$ and $v_j$ are directly connected. In this notation, a vertex can be regarded as a spatial entity in the study area, and the total number of entities ($n$) is finite.

A graph $G$ is connected if there exists at least one path between any two vertices. The length of a path, $L(v_i, v_j)$, is measured by the number of edges on the path between vertices $v_i$ and $v_j$. When $L(v_i, v_j) = 1$, it means that $v_i$ and $v_j$ are adjacent.

Besides the graph, we define two additional sets to describe the attributes of vertices and edges. The vertex attribute set is $A = \{a_1, a_2, \ldots, a_n\}$, where each component of $A$ is a set of $m$ attributes for a vertex in $V$. That is, $a_i$ is a vector of attributes for the $i$-th vertex. The basic attribute for an edge is the distance (or transportation cost) between the vertices at the two ends. We can extend this distance to a set $D$ that depicts the connections (e.g., distance or transportation cost) between any two vertices. $D$ is essential for solving many spatial problems. Intuitively, $D$ can be regarded as a matrix and $d_{ij} \in D$ is the measure of cost between any two vertices, $i$ and $j$, in $V$.

Given graph $G$ for a geographic optimization problem, a feasible solution to the problem is represented by another graph $G' = (V', E')$, where $V'$ is a set of vertices that forms a solution, and $E'$ depicts the spatial structure of vertices in $V'$. For selection problems, $V'$ is a subset of $V$. For partitioning problems, $V'$ contains the same vertices of $V$. However, those vertices that belong to one particular partitioning subdivision can be reasonably grouped as a unique subset. Therefore, we have $V' = \{U\}$, where $U$ is the subset that contains the vertices in one subdivision. It should be noted that a subdivision need not be spatially contiguous. Instead, in some cases, partitioning can mean categorization or classification, in which each spatial unit (represented as a vertex) is assigned an integer that indicates a particular class (see, for example, Bennett et al. 2004). For partitioning problems with spatial constraints, the structure of $V'$ or $\{U\}$ can be determined by $E'$.

Our goal in solving a geographic optimization problem is to find a solution $G^*$ that, without loss of generality, minimizes

$$F = (f_1, f_2, \ldots, f_k) \ , \tag{1}$$

where $F$ is a set of objective functions $(f_1, f_2, \ldots, f_k)$, and each objective function can be denoted as:

$$f_i : G \times G' \times D \times A \rightarrow \mathbb{R}, 1 \leq i \leq k \ . \tag{2}$$

# 3 Evolutionary Algorithms

The 1960s saw three similar, but independent, conceptual developments in Germany and the United States that let to the emergent field of evolutionary algorithms (EAs): evolutionary strategies (Rechenberg 1965), evolutionary programming (Fogel 1962), and genetic algorithms (Holland 1962). Active interaction among these groups began in the 1980s, which led to the formation of other new branches of research such as genetic programming (Koza 1992). The study of these algorithms collectively is called evolutionary computation and is now an area of intensive interdisciplinary research with a substantial literature that has been established during the past two decades (see Goldberg 1989; Bäck et al. 1997).

A typical and, to some extent, simple EA has the following components (see Forrest 1991):

- A population where each individual represents a solution to the problem.

- A scheme to randomly initialize and reproduce the population based on the fitness of individuals.
- A fitness function that is used to evaluate individuals according to the objective(s) of the problem.
- A set of evolutionary operators (including selection, recombination, and mutation) that is used to manipulate individual solutions.
- A set of parameters that defines the above features (e.g., population size and the probabilities of operators being invoked).

An EA can be regarded as a computer simulation of the natural evolutionary process and many EA components have been named using biological terminology. For example, an individual is also called a chromosome (a one-chromosome individual), and each feature (variable) in a solution can be called an allele. The procedure for a typical EA is outlined as follows:

Algorithm EA. *General procedure for an EA*
1 $t := 0$
2 Initialize population $G(t)$
3 REPEAT UNTIL termination criterion is satisfied
4 Evaluate each individual in $G(t)$
5 Select parents from $G(t)$ based on their fitness
6 Apply evolutionary operators to parents and produce offspring
7 $t := t + 1$

In Step 3, a maximum number of iterations is normally used as the termination criterion.

# 4 Design of EAs for Geographic Optimization Problems

The type of encoding used to represent feasible solutions to geographic optimization problems is determined by two conditions: the relation between $V'$ and $V$, and the characteristics of $E'$.

For $V'$, two kinds of relations with $V$ can be recognized:

- $V' \subset V$. $V'$ is a subset of $V$. Selection problems belong to this group.
- $V' = V$. In this case, the locations of $V'$ are identical to those of all vertices in $V$. Partitioning problems have this characteristic.

For relations between $E$ and $E'$, we know that $E$ defines the spatial relations among all vertices in $V$, and, that, equivalently, $E'$ confines the spatial relations among vertices in a solution. We can identify the following relations between $E$ and $E'$:

- $E' \subset E$. Selection or partitioning problems with spatial constraints belong to this category.
- $E' = \varnothing$. For some problems, spatial relations among vertices in a solution are not explicitly needed. To solve this type of problem, $E'$ is not required in the problem formulation, and

Table 1: Encoding for the $V/V'$ and $E/E'$ relations

| Code | Meaning | Relation | |
|---|---|---|---|
| | | $V$ and $V'$ | $E$ and $E'$ |
| $\Delta$ | Subset | $V' \subset V$ | $E' \subset E$ |
| $E$ | Equal | $V' = V$ | - |
| $\Phi$ | Empty | - | $E' = \varnothing$ |

we have $E' = \varnothing$. Selection or partitioning problems without spatial constraints are included in this category. Note that $E' = \varnothing$ means that $E'$ is not needed in the solution representation, but it does not imply that a spatial relation among vertices in $V'$ does not physically or logically exist.

A Greek letter is used to represent each type of relation discussed here (Table 1). With this notation defined, the encoding type used by each geographic evolutionary algorithm can be specified by a combination of two conditions. The first condition indicates the relation between $V$ and $V'$, and the second condition indicates the relation between $E$ and $E'$. Each combination is denoted by a string of two letters delimited by a slash (/):

- $\Delta/\Phi$: selection problems without spatial constraints,
- $\Delta/\Delta$: selection problems with spatial constraints,
- $E/\Phi$: partitioning problems without spatial constraints, and
- $E/\Delta$: partitioning problems with spatial constraints.

In addition, if we use an asterisk (*) to denote all possible conditions (or "don't care"), the following general problem types can be written as:

- $\Delta$/*: selection problems,
- $E$/*: partitioning problems,
- */$\Delta$: problems with spatial constraints,
- */$\Phi$: problems without spatial constraints, and
- */*: all geographic optimization problem types.

## 4.1 Encoding Strategies

For selection problems ($\Delta$/*), since $V' \subset V$, it is not necessary to record the location of each vertex in a solution. Instead, we can directly use the identification number of each vertex in the chromosome. In a *p*-median problem, for example, the chromosome is an array of integers (i.e., id's) with length equal to $p$. For partitioning problems ($E$/*), since $V' = V$, a string of $n$ integers can be used to represent feasible solutions; the integer value of the $i$-th element of the string indicates the subdivision to which the $i$-th spatial unit belongs.

For problems with spatial constraints (*/$\Delta$), the edge information must also be stored so that the spatial constraints can be effectively formulated and maintained. There are different

approaches to storing edge information. In previous research, edge information was explicitly recorded for each vertex in an individual solution of the EA (Xiao et al. 2002). For each individual solution, its data structure consists of a tuple $(v_i, N_i)$, where $v_i$ is a vertex, and $N_i = \{(v_i, v) \mid (v_i, v) \in E, v \in V\}$ is a set of neighbors for $v_i$. An individual solution, therefore, is represented by an array of $(v_i, N_i)$. While this data structure is effective, it is inefficient because whenever a solution is changed (i.e., some elements in $V'$ are changed), edge information for all vertices in $V'$ must be updated. The updating process must refer to $E$, which contains all edge information for $G$. The fact is, while $E$ is available, it is not necessary to keep edge information for each individual vertex redundantly. Consequently, the data structure for */$\Delta$ is still a string of integers. However, in contrast with problems without spatial constraints (*/$\Phi$), $E$ must be stored and made accessible for the entire EA.

## 4.2 Initialization Strategies

We can design an initialization algorithm based on an accretion procedure for selection problems ($\Delta$/*). In this algorithm, and others that follow, we use $p$ to denote the number of vertices to be selected, or the number of subdivisions to be partitioned. In addition, set $V'$ is always used to denote the vertices in a feasible solution.

Algorithm $I1$. *Accretion for selection*
1 $i := 0$
2 $V' := \varnothing$
3 $V_1 := V$
4 REPEAT UNTIL $i = p$
5 Randomly select a vertex from $V_1$
6 Add $v$ into $V'$
7 Update $V_1$ so that it only contains eligible vertices
8 $i := i + 1$

In Algorithm $I1$, a set of vertices ($V_1$) is maintained to contain vertices that can be added into $V'$ without violating any spatial constraint. In Step 7 of Algorithm $I1$, an eligible vertex means such a vertex that can be added into $V'$ without violating the spatial constraints. When no spatial constraint is required, $V_1$ will contain all unselected vertices in $V$ (i.e., $V_1 = V \setminus V'$).

We can develop a second initialization algorithm (Algorithm $I2$) to generate initial solutions to partitioning problems.

Algorithm $I2$. *Accreting for partitioning*
1 $i := 0$
2 $V' := \varnothing$
3 Randomly select $p$ vertices from $V$ and add them to $V'$
4 Assign the remaining $n - p$ vertices in $V$ to a subdivision

Step 3 of Algorithm $I2$ will result $V' = \{U_i | i = 1, ..., p\}$, where $U_i$ is a set for the $i$-th subdivision and it contains one and only one unique vertex, which serves as the seed of the subdivision. The specific procedure used in Step 4 varies from problem to problem. We will

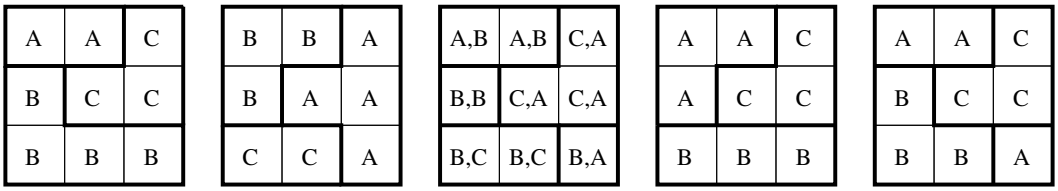



Figure 1: Overlay and repair for partitioning problems. Here, $p = 3$ and the letter in each cell indicates the subdivision ($A$, $B$, or $C$) of that cell. Each cell is indexed using row ordering: the cell on the upper-left corner is referred to as 1, while the cell on the lower-right corner is 9. (a) a hypothetical parent solution ($V' = \{\{1, 2\}, \{4, 7, 8, 9\}, \{3, 5, 6\}\}$; (b) another hypothetical parent solution ($V' = \{\{3, 5, 6, 9\}, \{1, 2, 4\}, \{7, 8\}\}$); (c) the result of overlay ($V_3'' = \{\{1, 2\}, \{3, 5, 6\}, \{4\}, \{7, 8\}, \{9\}\}$); (d) a possible result of a repair procedure when contiguity is required ($V' = \{\{1, 2\}, \{3, 5, 6, 9\}, \{4, 7, 8\}\}$); (e) a possible repair result when spatial constraints are not required ($V' = \{\{1, 2, 9\}, \{4, 7, 8\}, \{3, 5, 6\}\}$)

discuss how this can be achieved for political redistricting problems in Section 5. When no spatial constraint is required, Step 4 can be implemented in a random fashion in which each vertex is randomly assigned to a subdivision (see Bennett et al. 2004).

## 4.3 Design of Recombination Operations

The recombination operations reported in the literature share a similar behavior: the genetic make-up from two selected individuals (parents) is used to create a new individual solution (child). For geographic optimization problems, an overlay and repair approach can be used to recombine two individual solutions in an EA. In this approach, an overlay operation is conducted first to combine the vertices of two solutions $V_1'$ and $V_2'$ into a new set, $V_3''$. For selection problems, the overlay operation will result in a superset that contains vertices in both $V_1'$ and $V_2'$. For partitioning problems, however, a set of new distinct subdivisions will be generated by the overlay operation (Figure 1c). In both cases, the size of $V_3''$ (denoted as $|V_3''|$) normally will be larger than the size of either parent solution. A repair procedure is then needed to reduce the size of $V_3''$ to form a feasible solution.

The repair process can be implemented in multiple ways. For selection problems, one can adopt a dropping procedure in which vertices are removed from $V_3''$ until the number of vertices in $V_3''$ equals that of a feasible solution.

Algorithm $R1$. *Overlay and repair based on dropping*
1 $V_1'$ and $V_2'$ are selected from the population
2 $V_3'' := V_1' \cup V_2'$
3 $V' := \varnothing$
4 IF $|V_3''| = p$, THEN $V' := V_3''$ and STOP
5 REPEAT UNTIL $|V_3''| = p$
6 $V_1$ := FIND-MOVABLE-VERTICES($V_3''$)
7 Randomly select a vertex ($v'$) from $V_1$

8 Remove $v'$ from $V_3''$
9 $V' := V_3''$

In Algorithm $R1$, function FIND-MOVABLE-VERTICES($V'$) returns a set that contains all movable vertices in set $V'$. A vertex is movable if it can be removed from $V'$ without violating spatial constraints, if any. When no spatial constraint is required, all vertices in $V_3''$ will be moveable (in Step 6).

For partitioning problems with spatial constraints, a plausible approach to repairing $V_3''$ toward a feasible solution is through merging (Algorithm $R2$, see also Figure 1d).

Algorithm $R2$. *Overlay and repair based on merging*
1 $V_1'$ and $V_2'$ are selected from the population
2 $V_3'' := \{U | U$ is a set that contains all vertices in a subdivision after overlay$\}$
3 $V' := \varnothing$
4 IF $|V_3''| = p$, THEN $V' := V_3''$ and STOP
5 REPEAT UNTIL $|V_3''| = p$
6 Randomly select two subdivisions, $U_i$ and $U_j$, from $V_3''$
7 Merge $U_i$ and $U_j$ if doing so does not violate the constraints
8 $V' := V_3''$

For partitioning problems without spatial constraints, we can design a third recombination operation that assigns a cell to one of the subdivisions in its parent solutions. The assignment procedure can be carried out using a random number generator, or based on some user-specified rules. We call this algorithm $R3$, which is illustrated in Figure 1e.

## 4.4 Design of Mutation Operations

The design of mutation operations is based on a mechanism that changes the morphology and location of a solution. In this approach, an existing solution is modified by exchanging some of its vertices either with unselected vertices (for selection problems) or among different subdivisions (for partitioning problems). It is possible to develop a mutation operation that can be used for all types of problems (Algorithm $M1$, see below). Here a sets $V_2$ is maintained to contain the vertices that can be removed from $V'$ so that the resulting $V'$ does not violate spatial constraints. For problems without spatial constraints, $V_2$ will be identical to $V'$.

Algorithm $M1$. *Mutation*
1 $V_2$ := FIND-MOVABLE-VERTICES($V'$)
2 Randomly select a vertex ($v'$) from $V_2$
3 UPDATE($V'$, $v'$)

Function FIND-MOVEABLE-VERTICES($V$) has been discussed before. Function UPDATE($V'$, $v'$) repairs $V'$ by replacing $v'$ with a new vertex (selection problems) or assigning $v'$ to a new subdivision (partitioning problems); this function also guarantees that the spatial constraint, if any, not be violated. Algorithm $M1$ can be repeated a number of times during the EA execution.

### 4.5 Summary

Algorithms described above provide a guideline for the design and implementation of an EA to solve a geographic optimization problem. In general, an EA designed in this way can be denoted as a combination of five symbols:

$$v/e/i/r/m \ , \tag{3}$$

where $v \in \{\Delta, E\}$ indicating the relation between $V$ and $V'$,
$e \in \{\Delta, \Phi\}$ indicating the relation between $E$ and $E'$,
$i$ = the initialization algorithm,
$r$ = the recombination method, and
$m$ = the mutation method.

Using this notation, for example, we can specify the following geographic evolutionary algorithm:

$$\Delta/\Delta/I1/R1/M1 \ ,$$

which defines an evolutionary algorithm that uses encoding type $\Delta$/$\Delta$, and algorithms $I1$, $R1$, and $M1$ for initialization, recombination, and mutation, respectively. This EA can be used to solve a selection problem with spatial constraints.

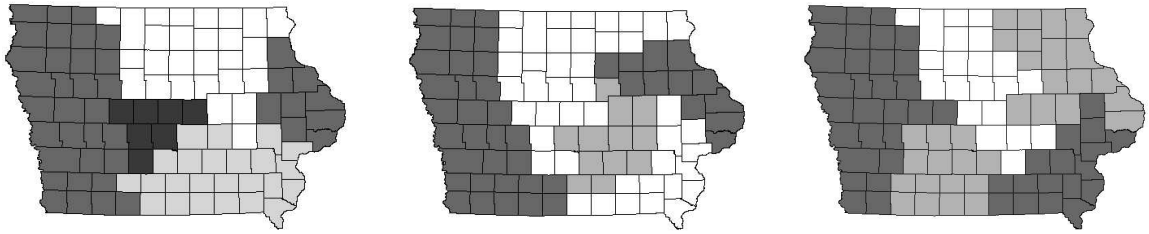
The graph representation discussed here can also be applied to guide the design of new types of spatial evolutionary algorithms that are not discussed in the previous sections. To incorporate new encoding types, additional algorithms may need to be developed to perform initialization, recombination and mutation operations.

## 5 An Application

Evolutionary algorithms have been used in a wide range of geographic optimizations. In this section, we focus on a partitioning problem with spatial constraints. In particular, we develop an evolutionary algorithm to solve political redistricting problems, in which a set of spatial units (e.g., counties or census tracts) must be subdivided into a number of contiguous regions (Williams 1995; Altman 1998). In this research, a redistricting plan is encoded as a string of $n$ integers, where $n$ is the number of total spatial units. Each element in the string is assigned an integer value, which ranges from 1 to $p$, where $p$ is the number of districts. We use $a_i$ to denote the population size in the $i$-th spatial units. The goal to solve this problem is to find a redistricting plan such that each district exhibits the ideal population size, $a^*$, which is computed as $\sum_{i=1}^{n} a_i/p$. We define the objective function as follows:

$$\min f = 100 \times \tfrac{1}{p \times a^*} \textstyle\sum_{i=1}^{p} |a_i - a^*| \ . \tag{4}$$

Overall, we can categorize the EA for redistricting as $E/\Delta/I2/R2/I2 + M1$. The initialization process is derived from Algorithm $I2$, which randomly selects $p$ vertices as seeds and then each spatial unit is assigned to its nearest seed. Here the distance between two spatial units (vertices) is measured by the length between them (i.e., number of edges on the path). Step 7 of Algorithm $R2$ merges $U_i$ and $U_j$ if they are adjacent. Two mutation operations are developed.



(a) $f = 0.022$ (b) $f = 0.069$ (c) $f = 0.008$

Figure 2: Redistricting plans generated by the EA (a and b), and the actual plan adopted for the 2000 congressional election (c)

The first merely applies the initialization operation (again, based on Algorithm $I2$) to create a new solution. The second mutation operation is based on Algorithm $M1$. In the implementation of Algorithm $M1$, function FIND-MOVEABLE-VERTICES in Step 1 randomly determines a district and returns the vertices on its border that can be removed from the district without violating the contiguity requirement; among these "moveable vertices", one will be selected and be switched to an adjacent district if doing so will not violate the contiguity (Step 3).

The EA outlined above was implemented using the C++ programming language on a Linux platform. We tested the EA using a case study of Iowa Congressional redistricting, based on Census 2000 data. According to Iowa Code section 42.4 and Iowa Constitution, a congressional district shall not vary from the ideal population by more than one percent, and counties shall not be split between more than one congressional district. To run the program, we set the population size to 20 and the total number of iterations to 2000. Based on this problem setting, the program was run on a Pentium M 1.4 GHz laptop and it finished in an average of 19.3 seconds. Two of the EA results are shown in Figures 2a and 2b. It can be noted that the EA can be used to generate high quality solutions, which satisfy the criteria required by law.

# 6 Conclusions

The rapid development of computing techniques during the past two decades has encouraged an optimistic view of the computational issues that face geographers (see, for example, Dobson 1983). However, scientific study should not wait for breakthroughs in computation technology. For computationally based research, it must be realized that the ultimate limitation on computation is the inherent complexity of the problem to be solved, rather than the speed of the computer system (Garey and Johnson 1979; Armstrong 1993). Developing new methods that overcome the shortcomings of existing methods has always been a motivation that leads progress in optimization research.

This paper addresses issues that are fundamental to geographic optimization in particular, and geographic information science in general, including conceptualization of geographic problems, spatial representation, and algorithm analysis; in this sense, it echoes geographers' recent interests in computational science (see Openshaw 1994; Fotheringham 1998; Armstrong 2000). In a broader view, this research provides the starting point of a quest toward the establishment of a unified framework for a variety of geographic optimization problems.

# References


Altman, M. (1998). *Districting Principles and Democratic Representation*. Ph. D. thesis, California Institute of Technology, Pasadena, California.

Armstrong, M. P. (1993). On automated geography! *Professional Geographer 45*(4), 440–441.

Armstrong, M. P. (2000). Geography and computational science. *Annals of the Association of American Geographers 90*(1), 146–156.

Armstrong, M. P., G. Rushton, R. Honey, B. T. Dalziel, P. Lolonis, S. De, and P. J. Densham (1991). Decision support for regionalization: A spatial decision support system for regionalizing service delivery systems. *Computers, Environment, and Urban Systems 15*, 37–53.

Bäck, T., D. B. Fogel, and Z. Michalewicz (Eds.) (1997). *Handbook of Evolutionary Computation*. New York: Oxford University Press/IOP.

Bennett, D. A., G. A. Wade, and M. P. Armstrong (1999). Exploring the solution space of semi-structured geographical problems using genetic algorithms. *Transactions in GIS 3*(1), 51–71.

Bennett, D. A., N. Xiao, and M. P. Armstrong (2004). Exploring the geographic ramifications of environmental policy using evolutionary algorithms. *Annals of the Association of American Geographers 94*(4), 827–847.

Brill, Jr., E. D. (1979). The use of optimisation in public sector planning. *Management Science 25*, 413–21.

Brookes, C. J. (2001). A genetic algorithm for designing optimal patch configurations in GIS. *International Journal of Geographical Information Science 15*(6), 539–559.

Church, R. L., D. M. Stoms, and F. W. Davis (1996). Reserve selection as a maximal covering location problem. *Biological Conservation 76*, 105–112.

Cooper, L. (1964). Heuristic methods for location-allocation problems. *SIAM Review 6*, 37–54.

Cova, T. J. and R. L. Church (2000). Contiguity constraints for single-region site search problems. *Geographical Analysis 32*(4), 306–329.

Crossland, M. D., B. E. Wynne, and W. C. Perkins (1995). Spatial decision support systems: an overview of technology and a test of efficacy. *Decision Support Systems 14*, 219–235.

Daskin, M. S. (1995). *Network and Discrete Location: Models, Algorithms, and Applications*. New York: John Wiley & Sons.

Densham, P. J. and G. Rushton (1996). Providing spatial decision support for rural public service facilities that require a minimum workload. *Environment and Planning B: Planning and Design 23*, 553–574.

Dibble, C. and P. J. Densham (1993). Generating interesting alternatives in GIS and SDSS using genetic algorithms. In *GIS/LIS '93*, Minneapolis, Minnesota, pp. 180–189. ACSM, ASPRS, AM/FM International, AAG, URISA.

Dobson, J. E. (1983). Automated geography. *Professional Geographer 35*(2), 135–143.

Dorigo, M., V. Maniezzo, and A. Colorni (1991). *Positive Feedback as a Search Strategy*. Dipartimento di Elettronica, Politecnico di Milano.

Fogel, L. J. (1962). Autonomous automata. *Industrial Research 4*, 14–19.

Forrest, S. (1991). Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. *Physica D 42*, 1–11.

Fotheringham, A. S. (1998). Trends in quantitative geography II: stressing the computational. *Progress in Human Geography 22*(2), 283–292.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: Freeman.

Ghosh, A. and F. Harche (1993). Location-allocation models in the private sector: progress, problems, and prospects. *Location Science 1*(1), 81–106.

Glover, F. (1989). Tabu search - part I. *ORSA Journal on Computing 1*(3), 190–206.

Glover, F. (1990). Tabu search - part II. *ORSA Journal on Computing 2*(1), 4–32.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Hakimi, S. L. (1965). Optimum distribution of switch centers in a communication network and some related graph theoretic problems. *Operations Research 13*, 462–475.

Han, J., M. Kamber, and A. K. H. Tung (2001). Spatial clustering methods in data mining: a survey. In H. Miller and J. Han (Eds.), *Geographic Data Mining and Knowledge Discovery*, pp. 188–217. London: Taylor and Francis.

Hof, J. and M. Bevers (1998). *Spatial Optimization for Managed Ecosystems*. New York: Columbia University Press.

Holland, J. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM 3*, 297–314.

Hopkins, L. D. (1984). Evaluation of methods for exploring ill-defined problems. *Environment and Planning B 11*, 339–348.

Hosage, C. M. and M. F. Goodchild (1986). Discrete space location-allocation solutions from genetic algorithms. *Annals of Operations Research 6*, 35–46.

Jaramillo, J. H., J. Bhadury, and R. Batta (2002). On the use of genetic algorithms to solve location problems. *Computers & Operations Research 29*, 761–779.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, Jr. (1983). Optimization by simulated annealing. *Science 220*, 671–680.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

Krzanowski, R. M. and J. Raper (1999). Hybrid genetic algorithm for transmitter location in wireless networks. *Computers, Environment and Urban Systems 23*, 359–382.

Miller, H. J. and S.-L. Shaw (2001). *Geographic Information Systems for Transportation: Principles and Applications*. New York: Oxford University Press.

Murray, A. T. and R. L. Church (1995). Measuring the efficacy of adjacency constraint structure in forest planning models. *Canadian Journal of Forest Research 25*, 1416–1424.

Openshaw, S. (1994). Computational human geography: towards a research agenda. *Environment and Planning A 26*, 499–505.

Osman, I. H. and J. P. Kelly (1996). *Meta-heuristics: Theory and Application*. Boston: Kluwer.

Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental, Problem Library translation No. 1122*. Royal Aircraft Establishment.

Reeves, C. R. (Ed.) (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell Scientific Publications.

Rushton, G. (1988). Location theory, location-allocation models, and service development planning in the third world. *Economic Geography 64*(2), 97–120.

Schilling, D. A., V. Jayaraman, and R. Barkhi (1993). A review of covering problems in facility location. *Location Science 1*(1), 25–55.

Scott, A. J. (1970). Location-allocation systems: a review. *Geographical Analysis 2*, 95–119.

Scott, A. J. (1971). *Combinatorial Programming, Spatial Analysis, and Planning*. London: Methuen & Co Ltd.

Simon, H. A. (1960). *The New Science of Management Decision*. New York: Happer & Row.

Vazirani, V. V. (2001). *Appromixation Algorithms*. Berlin: Springer.

Williams, Jr., J. C. (1995). Political redistricting: a review. *Papers in Regional Science 74*(1), 13–40.

Xiao, N., D. A. Bennett, and M. P. Armstrong (2002). Using evolutionary algorithms to generate alternatives for multiobjective site search problems. *Environment and Planning A 34*(4), 639–656.