Multi-objective Optimization for Establishing Geographical Boundaries and Associations Using Evolutionary Algorithms and Collaborative and Adversarial Intelligent Agents

by

Jonathan W. Lartigue

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 4, 2018

Keywords: Algorithms, Evolutionary Computing, Heuristic Search, Intelligent Agents, Machine Intelligence, Genetic Algorithms

Approved by

Dr. Richard Chapman, Associate Professor of Software Engineering
Dr. Gerry Dozier, Professor of Software Engineering
Dr. Dean Hendrix, Associate Professor of Software Engineering

Abstract

Evolutionary Algorithms (EAs) are useful in solving prohibitively large search problems in a limited amount of time. Where even the most efficient standard search algorithms can become unwieldy as the solution space expands, such algorithms can provide a near-optimal solution in a fraction of the time.

The application of both evolutionary and more traditional search algorithms to geographic districting problems is complicated both by the need to satisfy multiple, often contradictory, quality heuristics and also by the need for geographic connectedness. Because of this connectedness constraint, the ability of Evolutionary Algorithms to recombine, or evolve, existing solutions with the aim of creating even better solutions is hampered by the high likelihood of such changes producing an unconnected, and therefore invalid, solution.

This dissertation explores the difficulties involved in solving geographic districting problems with a contiguity constraint by first evaluating performance on real-world data for a small municipal area. Next, two key sub-algorithms — called Neighborhood Mutation and Local Repair — are introduced that can vastly reduce the incidence of invalid solutions in the population, and these sub-algorithms are objectively evaluated in a sandbox environment. Then, the full Evolutionary Algorithm, including Neighborhood Mutation and Local Repair, are executed on a statewide scale and its effectiveness is observed for independently satisfying multiple quality heuristics.

Finally, we assess the efficacy of the Evolutionary Algorithm to solve districting problems with multiple, sometime conflicting, quality heuristics. First, a traditional weighted multi-objective heuristic function is evaluated. Next, the heuristic evaluation task is divided among multiple, independent agents tailored for a specific quality attribute. These agents, which are formed into separate pools for each heuristic, then collaborative to produce an objective value for the entire solution.

Results show that for districting problems with a geographic contiguity constraint, a generic Evolutionary Algorithm can produce high-quality candidate solutions, but the progress of such algorithms is hindered by the increasing percentage of invalid solutions that are produced over time. We show that the Neighborhood Mutation and Local Repair algorithms are each highly effective in independently preventing invalid solutions from entering the population — and are even more effective when combined — in both sandbox and real-world scenarios, thus allowing the Evolutionary Algorithm to converge toward more nearly optimal solutions in a similar or even shorter amount of time.

Finally, we show that multiple, independent agents can separately evaluate the quality of part of a multi-objective heuristic and then collaboratively combine those assessments into a single fitness value that can perform comparably to a traditional, weighted multi-objective heuristic function.

Acknowledgments

The author expresses his sincere gratitude to Dr. Richard Chapman for his unwavering support throughout the author's graduate career. The author also thanks the members of the committee, Dr. Paul A. Harris, Dr. David Umphress, Dr. James H. Cross II, and Dr. Kate Lartigue.

For Courtney... without whose support this dissertation would not have been possible.

This dissertation was typeset using LaTeX and the Texpad and BibDesk applications for macOS. Statistical analysis and figures, unless otherwise indicated, are generated using RStudio and the R programming language [143].

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

"*Considering that we live in an era of evolutionary everything — evolutionary biology, evolutionary medicine, evolutionary ecology, evolutionary psychology, evolutionary economics, evolutionary computing — it was surprising how rarely people thought in evolutionary terms.*"

— Michael Crichton [51]

Evolutionary Computation (EC) in general, and Genetic Algorithms (GAs) in particular, are a "general, adaptable concept for problem solving" and are "especially well suited for solving difficult optimization problems" involving prohibitively large search spaces and a limited amount of time. [13, 15] Where even the most efficient standard search algorithms can become unwieldy as the solution space expands, evolutionary algorithms can provide a near-optimal solution in a fraction of the time.

The concept of Evolutionary Computation, and of Genetic Algorithms, stems from observations that, in nature, the processes of selection, reproduction, and mutation not only propagate a species from one generation to the next, but also improve the suitability, or adaptation, of a species to its environment over time. It is theorized, but widely accepted as axiom, that through this process the improvement and spread of desirable abilities or traits within a species, and also speciation itself, occurs.

In computer science, these concepts are embraced in the field of EC, which has the generalized aim of both effectively representing a problem within software and, over time, recombining potential solutions to that problem in such a way as to promote the longevity, or survival, of desirable solutions, or parts of solutions, to the problem. This is usually accomplished by defining a heuristic function that quantifies the suitability of a solution to the problem in such a way that it can be compared and ranked relative to other potential solutions. This quantity, usually called a fitness measure, is used to allow the relatively more-frequent selection of higher quality solutions as "parents" whose genetic makeup is then recombined or altered in varying ways to produce a new "child" solution that, it is hoped, will embody the more desirable parts of each parent and, therefore, become an even more fit, or suitable, potential solution to the problem.

This process of recombining parents to produce children leads to a new set of potential solutions, usually called a new generation, which then becomes the pool of available parents for this to repeat for as many generations as desired or necessary. Evolutionary and Genetic Algorithms typically continue to produce new generations until an arbitrary time limit expires, until a set number of generations has been reached, until a certain threshold of quality in potential solutions has been met, or until appreciable improvement in the quality of potential solutions is no longer occurring in subsequent generations. Once an algorithm terminates, the most suitable solution found is then returned as the best available solution to the problem.

1.1 Research Goals

In this research, we wish to explore the ability of Evolutionary Computation, specifically a modified genetic algorithm, to take a large geographic area and associated demographic data for that area and produce a set of geographic districts comprised of multiple smaller geographic parcels that are grouped according to any chosen criterion or criteria. This type of problem, typically referred to as a districting problem, poses unique challenges both in achieving multiple, often conflicting, attributes of quality as well as satisfying difficult geographic constraints.

For example, given a data set consisting of small geographic areas (e.g. U.S. Postal Service Zone Improvement Plan (ZIP) code parcels, census tracts, or similar small sections) and an associated set of demographic or similar information for each of those areas, we hope to show that Evolutionary Computation can generate any number of sets of districts of associated parcels based upon the characteristics of any desired demographic criterion or criteria. The criteria for these associations can be quite varied, and the manner of association of parcels can be such that associated subsets may be either geographically contiguous or separated.

The number of possible associations can quickly lead to an unrealistically large search space, even for small physical areas. For example, the area of Jefferson County, Alabama — excluding the surrounding metropolitan area — alone consists of 59 distinct zip codes. If we desired to partition these zip codes into five sets, based on an any arbitrary criterion, we would have a solution space of 5^{59} possible solution sets if geographic contiguity is not required, or more than 1.7×10^{41} possible solutions. If geographic continuity of a set is necessary, the number of possible solutions is much smaller, but still unrealistic to search. For an implementation with a statewide data set, a significantly larger number of zip codes — more than 800 in Alabama — vastly increases the search space. To partition this set into seven geographic regions would result in a search space of roughly 7^{811} and to partition it into 105 districts would result in about 105^{811} possible combinations, both of which are as near to infinity as practicable.

More accurately, the number of ways to partition a set of n elements into k non-empty sets is shown in the Stirling number of the second kind: [8][12][177][49]

$$S_2(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^i\Big(\frac{k}{(k-1!i!)}\Big)(k-i)^n \tag{1.1}$$

The application of traditional and evolutionary algorithms to the solving of districting problems is not new, but both traditional and evolutionary methods begin to degrade when multiple, independent quality heuristics are addressed simultaneously. Performance is impacted even more when geographic constraints such as district contiguity — which is difficult to quantify in a manner other than binary — are considered.

There are general similarities between many traditional search algorithms, which typically start with a candidate solution and then repeatedly iterate through its "neighborhood" by making small changes in hope of quality improvement, and evolutionary algorithms, which likewise operate on a a pool of many candidate solutions, modifying them over and over in hope of finding improved solutions. But both struggle with districting problems with contiguity constraints, where an otherwise good solution is considered invalid because of one or more geographic units in a district that are disconnected from the rest. In fact, with such a constraint in place, most random modifications to a valid solution would tend to produce an invalid solution as a result.

We aim to implement two approaches to mitigate the effects of discontiguous solutions in solving such districting problems with evolutionary algorithms. First, we intend to employ a preventative algorithm that uses an informed method of mutation that, compared to purely stochastic mutation, is less likely to produce a new candidate solution that is invalid. Secondly, we intend to create a remedial algorithm that will detect and attempt to correct minor contiguity flaws in a candidate solution with the aim of rendering that solution valid once again.

The problem of reconciling multiple independent quality heuristics is similar. In real-world districting problems, many considerations such as population equality, geographic compactness, preservation of communities, and socioeconomic equality, to name a few, must each be addressed, despite in many cases being incompatible with each other.

We attempt to address this by utilizing multiple intelligent software agents, each with a differing perspective of the simulated environment, to act as "stakeholders" for one or more quality heuristics for the search.

Research has shown that the use of intelligent software agents in evolutionary algorithms is becoming more common [156, 45], but many approaches to this regard the individuals in the population as agents themselves, rather than a truly independent software component. Agents, and multiple agent systems (MAS), have increasingly been associated with Evolutionary Algorithms, and although there has been wider acceptance of agent-based modeling in geospatial systems, limited research is available on the use of EAs in discovering geographic associations among discrete geographic units. There is even less research available on the use of multiple intelligent heuristic agents in informing the fitness functions of Genetic Algorithms.

Utilizing separate agents for independent evaluation of specific quality attributes allows each to inform, individually, on the quality or suitability of a solution in terms of that specific attribute and, collectively, on all attributes. Therefore, the quality measure of any candidate solution will no longer be based upon a single calculation, but upon the collection of independent calculations from multiple independent agents with different heuristics particular to the interests of each agent. When considered together, the agents can produce a single quality measure for each candidate solution that can be used for ranking purposes, which is necessary in Evolutionary Algorithms during selection and reproduction operations. Thus, the population as a whole will tend to evolve and produce candidate solutions with fitness values that, on average, are increasingly suitable for most or all of the intelligent agents.

We further expand this this concept by implementing two or more "pools," or "agencies," each of which is composed of a collection of agents that — within that pool, at least — have similar perceptions of the simulated environment and quality goals for candidate solutions. However, since each separate pool is concerned with its own quality metric, one pool's perceptions may conflict with another pool. Thus, when one pool would tend to find all or part of a candidate solution appealing, the competing pool may not necessarily rate all or part of that same solution as highly. The total quality of any candidate solution then depends on the combined quality estimations of each of the agents.

When two or more separate pools of agents evaluate the quality of a candidate solution, some of these quality measures will be adversarial in nature to the criteria favored by other pools. But at the same time, the members within the same pool will share some in common a set of similar, collaborative criteria. Therefore, any given candidate solution will be assigned a quality value by competing pools of both *collaborative and adversarial intelligent agents*, in which the separate pools may be adversarial in their goals, but the members within those pools are collaborative in theirs.

The matter then becomes one of how to resolve the two competing fitness values assigned by each pool. If we presume that the pools are adversarial in nature, then in many cases as one pool would tend to assess a relatively high fitness value for a candidate solution, we would expect the competing pool to assess that same candidate solution relatively lower. In this situation, we hope to reduce these dual values to a single fitness score that reflects a measure of agreement — or disagreement — between the two pools. Thus, the modified fitness value that is used to order the candidate solutions in the population will come to reflect the degree to which both pools of agents agree on the suitability of a solution. Through this approach, we hope to show that the most mutually agreeable solutions rise to the top and propagate into subsequent generations, eventually producing a candidate solution that both pools of agents find to be the most nearly agreeable or, conversely, the least disagreeable.

1.2 Hypotheses

Hypothesis 1 Evolutionary Algorithms can successfully produce increasingly improving solutions for districting problems with a geographic contiguity constraint and one or more independent criteria for heuristic evaluation.

Hypothesis 2 The negative impact of a geographic contiguity constraint on an evolutionary algorithm can be mitigated by modification of destructive operations, such as recombination and mutation, to limit the likelihood of infeasible genetic characteristics entering the population.

Hypothesis 3 The negative impact of a geographic contiguity constraint on an evolutionary algorithm can be mitigated by the implementation of a mechanism to detect and repair infeasible portions of the genetic string.

Hypothesis 4 Independent Intelligent Agents, each with differing perceptions of the simulated environment and goals for candidate solutions, can work together to provide fitness estimates that will allow the population to evolve increasingly more fit solutions.

> **Hypothesis 4-A** A set of Independent Intelligent Agents can disagree in the relative fitness of candidate solutions, yet cooperatively inform on the fitness to produce a ranking measure that will allow an evolutionary algorithm to evolve the population over time.
>
> **Hypothesis 4-B** Given a set of Independent Intelligent Agents informing on the fitness of candidate solutions in a genetic algorithm, the presence of candidate solutions that are infeasible to one or more agents, but not all agents, can still prosper and evolve into candidate solutions that are feasible to all agents.
>
> **Hypothesis 4-C** Given two or more pools of independent agents, each of which is comprised of a group of agents that have similar perceptions of the simulated environment and goals for candidate solutions, and each of which has differing perceptions and goals from the other pools, these pools of agents can work collaboratively within the pool and adversarially (i.e. externally) between pools to produce a ranking measure that will allow for the operations of an Evolutionary Algorithms to positively evolve the population over time.

1.3 Research Overview

The first phase of this research focuses on a traditional Genetic Algorithm working with a small set of geographic parcels — less than 60, representing a single county. This phase centers on the construction of the Genetic Algorithm itself, the representation or encoding of the geographic parcels in software, the development of a meaningful fitness measure, the incorporation in the design of appropriate patterns to support later phases of research, and establishing a baseline of performance for the algorithm and its heuristic measures.

Next, the research focuses on objectively evaluating the performance of a modified Genetic Algorithm in a controlled, sandbox environment with simulated data that is of a large enough size to contain a number of geographic parcels comparable to what would be found when performing a districting problem on a typical U.S. state. The algorithm's performance is objectively assessed in terms of population size, selection method, recombination effectiveness, and mutation effectiveness. Two algorithms are implemented for reducing the incidence of invalid solutions in the population: a Neighborhood Mutation algorithm that attempts to prevent the creation of invalid solutions; and a Local Repair algorithm that attempts to correct small discontiguity errors that do occur. The effectiveness of these algorithms are independently assessed using the sample data set and sandbox.

Third, the algorithm is tailored based upon previous results and employed on a state-wide scale using real-world demographic and geospatial data for the state of Alabama — Encompassing more than 640 geographic parcels — and tested with a number of differing quality heuristics.

Finally, the algorithm's performance is tested with a multi-objective heuristic function and then using pools of software agents, each of which has its own quality criterion, to together assess each of the quality criteria.

Chapter 2

Literature Review:

Evolutionary Computing

"*(I resolved to) conduct my thoughts in such an order that, by commencing with objects the simplest and easy to know, I might ascend, little by little, and as it were, step by step, to the knowledge of the more complex.*"

— Rene Descartes [59]

2.1 The Beginnings of Evolutionary Computation

Through the processes of neo-Darwinism evolutionary theory — reproduction, mutation, competition and selection — the best individuals in a population are presumed to persist and bear offspring from generation to generation, thus preserving their traits and abilities, and more importantly the genetic material that makes them stronger, better, or otherwise superior to the rest of the population [53].

It can be presumed that, through attrition, those individuals that perform poorly are eliminated from the population by either physical death or by genetic death (i.e., not reproducing) and those that are fit and possess desirable characteristics survive and reproduce. It is through this process that the population as a whole evolves its fitness and characteristics.

Fogel writes that Darwinian evolution is “intrinsically a robust search and optimization mechanism” [68]. Artificial intelligence methods can leverage this premise to solve complex problems [152]. Evolutionary Computation (EC), and more specifically Genetic Algorithms, applies this biological theory to the solving of complex problems through simulated evolution. When introducing Evolutionary Computation, Fogel succinctly describes evolution as a “two-step process of random variation and selection” [66].

To accomplish this, a set of anthropomorphic or otherwise representative objects is created to model, in software, a candidate solution to a complex, real-world problem. These candidate solutions comprise the population and are considered to simulate some of the capabilities of a biological organism, such as the ability to combine, or mate, with another instance and, in doing so, exchange genetic material to produce a new instance: an offspring or child [152].

The process of mating and reproduction allows the creation of any number of distinct, subsequent generations. By selecting the individuals in the population that are deemed to be the most desirable — and therefore are presumed, but not guaranteed, to possess the more desirable traits — as parents for the next generation, it is hoped that the average quality of the population can be improved over time [66]. The advantages of Evolutionary Computation in solving difficult optimization problems include the simplicity of this approach, the the ability of these algorithms to respond to changing circumstances, and flexibility in representing different problems [68].

2.1.1 Origin of Evolutionary Computation

Modern implementations of Evolutionary Computation (EC), which now encompasses the field of study concerning all types of evolutionary inspired algorithms, has its roots in three very similar, but distinct and independently developed approaches: Genetic Algorithm (GA)s, Evolutionary Programming (EP), and Evolutionary Strategies (ES) [13, 15][65][68].

A fourth related approach, called Genetic Programming, emerged later but is now considered part of the domain of Evolutionary Computation [114].

The origins of these are variously attributed to several individuals, and particularly among them George E. P. Box and Alex S. Fraser, both of whom published the initial concepts that would later lead to the broader field of Evolutionary Computation in general and Genetic Algorithms in particular [73] [27].

The inspiration for both of these efforts came, in part, from the much earlier work of Sewall Wright, whose work in population genetics, specifically the relationship between genotypes and pheontypes, would lead to his 1932 description of "adaptive landscapes" [183], which are a metaphor to communicate the complexities of additive and epistatic effects on fitness in a population [165]. He developed an early mathematical theory of evolution, modeling how the frequencies of alleles in a population could vary based on "pressures" such as natural selection, mutation, speciation, and migration [98].

The work in the late 1950s of Box on what was termed Evolutionary Operation (EVOP) was one of the first attempts to introduce small changes to a problem or system and, over time, determine which improvements tend to increase the suitability or quality of the problem or system being modified [27]. Box's approach, which essentially consisted of a series of successive experiments both in the laboratory and on the factory floor, envisioned a "yield surface," similar to Wright's "adaptive landscapes," in which the maximum yield of a chemical manufacturing process could be envisioned as a "maximum mound" on a two-dimensional plot of time and percentage yield [28]. Box noted that the optimum conditions found experimentally in the lab sometime differed from the optimum conditions for maximum yield in large scale manufacturing, and he proposed small variations in the manufacturing parameters that, while not significantly affecting the suitability of the final product, would allow the investigation of neighboring areas of the "yield surface" to find improvements. Box wrote that the two essential features of the evolutionary process were *variation* and *selection of 'favourable' events* [27, 84].

At about the same time, A.S. Fraser was conducting the first experiments simulating genetic systems using digital computers and evaluating the effects of selection [73]. His early efforts studied diploid organisms represented by binary strings, each bit of which represented a specific dominant or recessive allele and explored simulated reproduction of organisms using a multi-point crossover operation, both of which techniques would later form the core concepts of Genetic Algorithms [67]. His works are notable, in particular, for introducing the Monte Carlo method for selection of parents in genetic systems and for enumerating a variety of techniques for manipulation of the performance of genetic simulations [67].

Evolutionary Programming (EP), pioneered by Lawrence Fogel, and refined by Burgin, Atmar, his son David Fogel, and others, has its roots in early approaches to demonstrate artificial intelligence by evolving finite state machines used in symbol prediction [9][13][34][69][70][71]. The Evolutionary Programming concept matured into an approach for solving optimization problems that incorporates a population of candidate solutions, the replication and mutation of solutions into new candidate solutions, and the assessment of the fitness of candidate solutions to determine if an individual solution is allowed to remain in the population [35].

Unlike Genetic Algorithms, which will be described in depth below, Evolutionary Programming typically does not restrict the manner of representation of the problem domain, the number of offspring produced by each member of the population, or employ crossover of genetic material from multiple parents to an offspring. Although a form of selection exists in Evolutionary Programming, it is generally used not to determine which individuals will be allowed to reproduce, but which offspring will be allowed to survive [35].

Evolutionary Strategies (ES) were introduced at the Technical University of Berlin by Rechenberg, Schwefel, and others beginning in the early 1970s, again with the goal of addressing optimization problems “strategically” instead of with more traditional gradient or hill-climbing strategies [145][158][159]. That was done by applying stochastic changes, following the example of mutation in the natural world, to the parameters of a shape problem.

As originally proposed, Evolutionary Strategies relied on a population of one and mutation operations to conduct the search for other candidate solutions [117]. This approach was augmented with a selection operation, and systematic testing of other Evolutionary Strategies approaches was begun, experimenting with variations in the number of parents used to produce offspring, the number of offspring produced by each parent, narrow and wide variation through mutation, and other factors [36].

Evolutionary Strategies permitted some of the first detailed, empirical analysis of the performance of Evolutionary Computation, with insights gained in population convergence, desirable mutation ratios, and other areas [36]. The concepts of Evolutionary Strategies would eventually spill over into other areas of Evolutionary Computation, and would particularly influence the broadening of Genetic Algorithms beyond their canonical form. Today, Evolutionary Strategies and Genetic Algorithms share many features [117].

The subfield of Genetic Algorithms (GAs) was developed by John Holland, whose 1975 book "Adaptation in Natural and Artificial Systems" is generally considered to be among the first definitive works on the subject and which and laid the foundation for most future discussion and experimentation on GAs [126]. Holland's book, which was the culmination of his earlier work [89], arose from exploration of adaptation in nature and how it could be used in computer systems. This work established much of the terminology and defined the basic operations of traditional GAs, codifying the core concepts of crossover, mutation, and inversion [92]. He presented the theoretical foundations of modern GAs, and also defined the concept of a "schema," or representations of genetic substrings that represent portions of a chromosome.

Holland examines adaptation as it applied to any system, and suggests that adaption is the "progressive modification of some structure or structures." The precise mechanism by which this is done is not as important as the concept that adaptation "designates any process whereby a structure is progressively modified to give better performance in its environment" [92].

He refers to an "adaptive plan" that consists of the structures in the system and the mixture of operators acting upon it. The plan's "domain of action" is the set of all structures that can be obtained by applying all possible combinations of operator sequences repeatedly. Furthermore, he states that it is the purpose of an adaptive plan to provide structures "which perform well (i.e. are more 'fit') in the environment confronting it" [92].

Holland describes three major components in the adaptive process: the environment, E, of the system undergoing adaptation; the adaptive plan, τ, by which the structures are modified; and a fitness measure, μ, of the structure's performance in its environment [91].

Initially, an adaptive plan has no knowledge of what constitutes a fit structure in its environment. However, over time a plan tests the performance of many different structures in the environment. Because one environment may differ from another, the current environment E is a subset of all possible alternative environments $E \in \epsilon$. For each different environment, the measure of fitness may be different, so each environment has its own performance measure μ_E [91].

Holland underscores the importance of evaluating the fitness of a structure and notes that each problem domain for adaptive systems is "defined as much by its performance measures as by its structures and operators." The set of these operators, Ω, can be applied successively in any order to produce a sequence of modifications to the set of structures, α, over time. Holland summarizes the challenges of the adaptive process by stating "that the organization of α, the effects of the operators Ω upon structures in α, and the form of the performance measure μ_E all affect the difficulty of adaption" [91].

The obstacles to adaptation include the very large — and often impossibly large – set α and the length and complexity of representative structures that obscure which specific subcomponents of the structures are most relevant for good performance. Additionally, the performance measure μ_E is often a complicated, non-linear function with multiple parameters that results in an uneven landscape comprised of local optima and discontinuities [91].

Finally, performance measures can vary over time and location, so fit structures may only be relevant at certain places and times.

There are biological analogues to underscore the difficulty of determining exactly which part of a structure is most important for all or part of a high performance measure for that structure. Holland points out that "every organism is an amalgam of characteristics determined by the genes in its chromosomes" and that each of these genes has multiple forms or alternatives, known as alleles. In vertebrate species there are approximately 10,000 genes, and if each gene has only two possible alleles, then the number of possible genetic combinations is $2^{10,000}$ [91].

Piglucci notes that the vastness of possible genetic combinations was apparent to Sewall Wright [183]. "Wright quickly calculated that reasonable assumptions about the number of allelomorphs possible even with (modest) figures was orders of magnitudes higher than the number of particles in the universe" [137, 592].

However, in biology, a single gene is often not independent from others in affecting characteristics of an organism. Instead, a single gene may have an impact on multiple otherwise independent characteristics or, conversely, a single characteristic may depend on the interactions of more than one gene. This phenomenon is referred to as epistasis and demonstrates that the effect of one gene at a particular location in a chromosome is dependent on the presence of one or more other genes. This tendency greatly increases the complexity of the system and the presence of multiple alleles affecting the observed characteristics of the system, known as the phenotype, depends on epistatic effects. Because of this, Holland writes, "there is no simple way to apportion credit to individual alleles for the performance of the resulting phenotype" [92].

Holland's adaptive plans concept, and similar work by his students [38][93], was applied to optimization problems by Kenneth De Jong in his 1975 doctoral dissertation under Holland. De Jong assessed the performance of an adaptive algorithm versus stochastic search in parameter optimization problems using a variety of functions of varying geometry, from simple three-dimensional parabolas to more complex surfaces with multiple maxima and minima [57]. His work provided an empirical study of adaptive algorithms, assessing variations in population size, mutation rate, crossover rate, and elitism. These same concepts are still central to modern Genetic Algorithm theory and practice. He would also plant the seeds for future work in GA theory, including multi-point crossover, diploid representations, and gene dominance [57][58].

Today, despite the relatively independent development of Evolutionary Programming, Evolutionary Strategies, and Genetic Algorithms, the distinctions between these different representations and approaches have sufficiently blurred to the point that "it now makes little sense...to speak of these originally different methods as being currently disparate" [68, 12].

2.1.2 General Evolutionary Computation Algorithms

All forms of Evolutionary Computation (EC) algorithms share the same general characteristics. The first of these is a population of arbitrary size containing candidate solutions, each of which is an encoding of the problem to be solved, and each of which has an objective function that quantifies the quality, desirability, fitness, or otherwise "goodness" of an individual candidate solution. The second characteristic is a means of recombining the characteristics, encoding, or genes of individuals in the population to either modify existing individuals or to create new individuals. The final characteristic is a means of selection that is typically used to determine either which individuals are chosen for recombination or which children are allowed to enter the population and survive. Thus, over successive iterations of each of these processes, it is hoped the overall quality of the population increases.

One can summarize this by defining a general formula for all evolutionary algorithms: [63, 2-3][136]

$$p_{t+1} = s(v(p_t)) \tag{2.1}$$

Algorithm 2.1: A general procedure for an Evolutionary Algorithm. [184][185]

```
Data: Given an empty population $G$ of candidate solutions at time, or generation, $t$
Result: A population $G$ containing one or more highly fit candidate solutions
1  begin
2      $t \leftarrow 0$
3      Initialize population $G(t)$
4      repeat
5          Evaluate each individual in $G(t)$
6          Select parent(s) from $G(t)$ based on fitness
7          Apply evolutionary operators to parent(s) to produce offspring
8          $t \leftarrow t + 1$
9      until termination criteria satisfied;
10 end
```

where p_t is the population at any given time t, $s(x)$ is the selection operation for choosing individuals from the population for reproduction, and $v(x)$ is the genetic variation operation, which is responsible for recombination of genes from parents to produce offspring. This can be expressed in pseudocode as shown in Algorithm 2.1.

Evolutionary Computation is merely one of many approaches for conducting optimized heuristic search, and some might argue that they are a logical extension to existing non-evolutionary optimization approaches. But they are not necessarily always the best type of algorithm for solving such problems. The “No Free Lunch” theorem, offered by Wolpert and Macready, holds that no single algorithm or method is a “silver bullet” in that it will perform equally well on all types or classes of problems. That is, for any algorithm or method that excels in solving one specific class of problem, there exists another, different class of problems for which it will perform less optimally than other algorithms or methods [181].

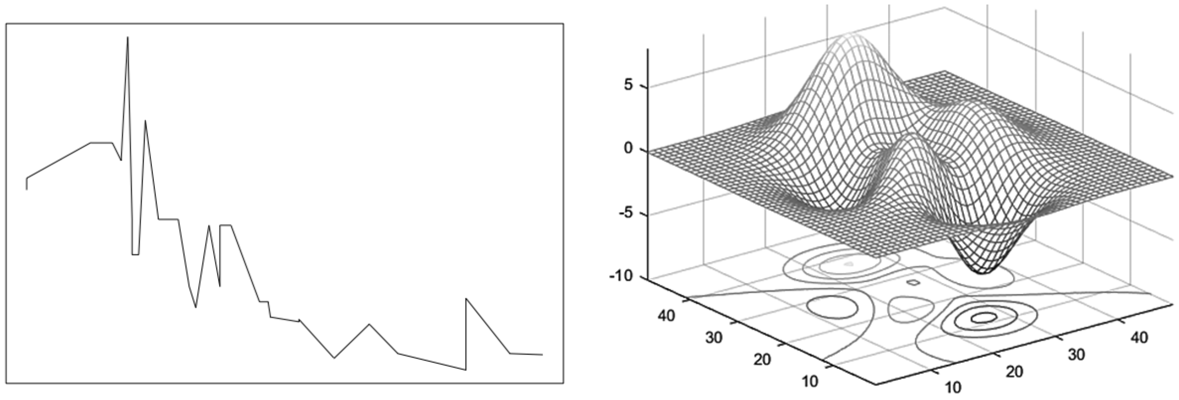

Figure 2.1: A sample two-dimensional and three-dimensional rendering of hypothetical solution spaces depicting multiple maxima (peaks) and minima (valleys) that can pose difficulties for some optimization algorithms.[1]

2.2 Traditional Search Methods

When discussing optimized search, a "search space" refers to the set of candidate solutions to a problem which, when mapped into an n-dimensional space or hyperspace, can be said to be separated from each other by some distance metric [126, 6]. This search space encompasses all possible solutions to a particular problem, and for most meaningful problems would typically include a large number of independent variables; a search problem with n independent variables would map to an n-dimensional search space.

For illustrative purposes, we can simplify this visualization as two- or three-dimensional plot depicting a contiguous line or surface onto which all possible solutions are mapped *(see Figure 2.1)*. Although each of these usually has a single, globally maximum value (represented by the tallest peak), we can also observe multiple local maxima (peaks) and local minima (valleys) in the solution spaces.

The concept of a search space is similar to that of a "fitness landscape," or an "adaptive landscape," originally described by Sewall Wright [183] in reference to biological genetics and efforts to measure both the "distance" between two genotypes and the tendency of

[1] Three-dimensional image courtesy of The MathWorks, Inc. [168]

natural selection to move a population over time to a “peak” of relatively high fitness in the landscape [126]. Box refers to essentially the same concept as a “response surface” [26].

The distance between two individuals in a fitness landscape can be expressed in terms of the number of differing genes between two genomes and the difference between their relative fitnesses. Wright offers a two-dimensional plot of the fitness landscape to visualize a population as “genotypes...packed side by side ... in such a way that each is surrounded by genotypes that differ by one one gene replacement” [137].

For example, given two genomes of length l, and a real-valued fitness measure, a fitness landscape can be envisioned as an $l+1$ dimensional space in which each gene value is plotted along the first l axes and the fitness is plotted along the $l+1$th axis. The number of differing genes, referred to as the Hamming distance for purely binary-coded chromosomes, is the number of genes that have differing allele values between the two. Therefore, in the fitness landscape, the distance between the two individuals is a vector combination of the number of differing genes and the difference in the fitness values between the two individuals [126].

Wright illustrates that such landscapes form hills and valleys of relative high and low fitness, respectively, and that the process of adaptation through natural selection tends to move a population toward a nearby peak of relatively high fitness. But Wright, writing in 1932, foresaw a problem that would prove to daunt early Evolutionary Computation researchers, and which remains a challenge to this day: “The problem of evolution as I see it is that of a mechanism by which the species may continually find its way from lower to higher peaks in such a a field” [183, 358-359]. He writes further: ”.. there must be some sort of trial and error mechanism on a grand scale by which the species may explore the region surrounding the small portion of the field which it occupies. To evolve, the species must not be under strict control of natural selection.” [183, 359]

There have been a large number of optimization algorithms — both domain-specific and generalized — developed to find either the absolute maximum value or near-maximum value in such search spaces. However, absent a brute force exhaustive search of all possibilities,

Algorithm 2.2: A generalized "steepest-ascent" hill-climbing algorithm [126, 96].

```
Data: A binary encoded problem with data length $l$
Result: The highest quality solution found
begin
    $C \leftarrow$ Random candidate solution
    $C_{best} \leftarrow C$
    repeat
        repeat
            foreach bit $i \in (0..l)$ do
                Mutate bit $i$ to produce $C'$
                Evaluate the fitness $f(C')$
                if $f(C') > f(C_{best})$ then
                    $C_{best} \leftarrow C'$
                end
            end
        until $C_{best}$ is unchanged;
        $C \leftarrow$ Random candidate solution
    until termination criteria are met;
    return $C_{best}$ as the highest hilltop found
end
```

many of these algorithms generally start with a potential value or solution and then attempt to explore nearby values iteratively in order to find increasingly better values or solutions. Local maxima and minima can represent challenges to such algorithms, which can become stuck at a locally high value that is inferior to globally optimum value.

2.2.1 Hill Climbing and Gradient Ascent

A common approach to optimization problems are greedy neighborhood-search algorithms such as gradient ascent (or descent) and hill climbing, both of which attempt to maximize (or minimize) a function or heuristic.

A typical hill climbing algorithm, also called an iterative improvement algorithm, attempts to reach an optimum solution by iteratively improving a single solution with small modifications to its parameters. Generally, it can be said that a hill climbing algorithm evaluates nearby solutions in the search space searching for a better solution than the current one, selecting only non-declining values for the next move. In a three-dimensional context, one can visualize a series of iterative improvements in a solutions that, over time, move the solution up a slope one small step at a time until, eventually, reaching a peak. The most common approach is known as steepest-ascent hill climbing *(see Algorithm 2.2)*, in which all neighbors are evaluated and the most-improved is selected, but variations exist such as next-ascent hill climbing, in which the first neighbor found with improvement is selected, and random-mutation hill climbing [126, 96-98]. There are many other approaches to hill-climbing, including: exhaustive neighborhood evaluation before selecting the best move; stochastic hill climbing, which selects randomly from all increasing moves; a "first improvement" strategy, which chooses the first potential move that results in improvement; neutral-move policies, which enable movement across relative flat landscapes; and others [120][121].

Gradient ascent (or descent) algorithms are similar in concept to hill-climbing algorithms with the caveat that gradient ascent (descent) always chooses the move with the steepest ascent (descent) at any given moment. The movement in the solution space can be seen as choosing the steepest ascent or descent along the slope of the curve at the current location. In this manner, gradient descent always seeks the greatest immediate improvement from one step to the next [45]. Gradient ascent (descent) algorithms are known to have difficulty climbing ridges (or valleys), and such algorithms tend to slowly "zig-zag" along these ridges or valleys in an attempt reach a maximum or minimum value [117].

Problems arise in both hill-climbing and gradient ascent (descent) algorithms when the solution space contains multiple peaks or valleys. Depending on the starting location of the algorithm, it can tend to climb to the top of the local peak or saddle point and stop, regardless of whether that peak represents a global optimum or merely a local one [45][68]. Various methods have been employed to mitigate this problem, including running such algorithms multiple times with randomized starting locations, breakout functions which will allow the algorithm to potentially jump from its hill to a nearby one, algorithms that allow temporarily exploring an inferior solution, and other techniques.

Algorithm 2.3: A generalized simulated annealing algorithm [77].

```
Data: A random starting solution $S$, a constant cooling factor $\alpha = 0.9$, a minimum
      temperature $T_{min} = 0.00001$, and a number $k$ of neighbors to search before
      “cooling” the algorithm.
Result: A lower-cost solution $S'$
begin
    $S_{old} \leftarrow S$
    $T \leftarrow 1.0$
    while $T \geq T_{min}$ do
        foreach $i \in (1..k)$ do
            $S_{new} \leftarrow \text{neighbor}(S_{old})$
            $p_a \leftarrow e^{\frac{S_{new} - S_{old}}{T}}$
            if $p_a \geq randomInt(0..1)$ then
                $S_{old} \leftarrow S_{new}$
            end
        end
        $T \leftarrow T \times \alpha$
    end
    $S' \leftarrow S_{old}$
    return $S'$
end
```

2.2.2 Simulated Annealing

Search by simulated annealing is based on the metallurgical concept of annealing, in which a metal is heated to a temperature that exceeds its recrystallization temperature and then slowly cooled, thus allowing atoms in the structure to migrate, resulting in a change in the malleability, ductility, or hardness of the metal [180]. The computational method of simulated annealing employs this principle by defining a “heat” factor of the system in which the search is being conducted. When the heat of the system is high, the solution or value is allowed to fluctuate greatly between one iteration and the next, allowing the search to make relatively large jumps from one potential solution to the next. When the heat of the system is low, the solution is allowed much smaller perturbations [117][128].

Generally, such an algorithm starts off with a very high heat level, allowing wide jumps around the search space and thus greater exploration of the search area. Then, as the system

Algorithm 2.4: A generalized Tabu search algorithm [31].

```
Data: A starting solution $S$ and a maximum Tabu list size $Tabu_{max}$
Result: The best solution $S_{best}$ found
begin
    $S_{best} \leftarrow S$
    while Stopping conditions not reached do
        Set of candidate solutions $C \leftarrow \emptyset$
        foreach $S_{candidate} \in Neighbors(S_{best})$ do
            if $S_{candidate} \notin Tabu$ then
                $C \leftarrow C + S_{candidate}$
            end
        end
        $C_{best} \leftarrow max(C)$
        if $Cost(C_{best}) \leq Cost(S_{best})$ then
            $S_{best} \leftarrow C_{best}$
            $Tabu \leftarrow Tabu + S_{best}$
            while $Size(Tabu) \geq Tabu_{max}$ do
                $Tabu \leftarrow Tabu - \mathrm{Oldest}(Tabu)$
            end
        end
    end
    return $S'$
end
```

is slowly cooled, the magnitude of the jumps is reduced and the solution begins to oscillate toward a maxima or minima that is hoped to be optimal or near-optimal *(see Algorithm 2.3)*. The fluctuations throughout wide areas of the search space as the heat is relatively high are the means by which simulated annealing attempts to overcome the problem of local maxima and minima [2][74][106][128].

2.2.3 Tabu Search

Tabu Search (TS), first proposed by Fred Glover in 1977, derives its name from an alternate form of the word "taboo," is quite similar to hill-climbing or gradient descent algorithms, with the exception that the algorithm is not deterministic [79]. Instead, the algorithm maintains a "memory" of previous states or values and the relative desirability, or more accurately the undesirability, of those states. This information is stored in a "tabu list"

Algorithm 2.5: A generalized particle swarm optimization algorithm [31][123].

```
Data: A population size $n$
Result: The best particle position found
begin
    $P \leftarrow \emptyset$
    $P_{global\_best} \leftarrow \emptyset$
    foreach $i \in (1..n)$ do
        $p \leftarrow$ new Particle()
        $p_{velocity} \leftarrow$ randomVelocity()
        $p_{position} \leftarrow$ randomPosition()
        $p_{pos\_best} \leftarrow p_{position}$
        $P_i \leftarrow p$
    end
    repeat
        foreach $p \in P$ do
            $p_{velocity} \leftarrow$ updateVelocity($p_{velocity}, P_{global\_best}, P_{best}$)
            $p_{position} \leftarrow$ updatePosition($p_{velocity}, P_{global\_best}, P_{best}$)
            if $f(p) > f(P_{best})$ then
                $P_{pos\_best} \leftarrow p_{position}$
            end
        end
        $P_{global\_best} \leftarrow \max(P)$
    until $iteration \geq iteration_{max}$ or quality threshold reached;
    return $P_{global\_best}$
end
```

of previous states in the local neighborhood [117]. As the search progresses, the algorithm computes future states or values by taking into account previously searched undesirable states or values *(see Algorithm 2.4)*. The algorithm will generally endeavor to avoid previously visited states and avoid undoing previous changes.

2.2.4 Particle Swarm Optimization

Particle Swarm Optimization, like Genetic Algorithms, is a multi-path search technique that investigates many potential solutions simultaneously. Unlike Evolutionary Algorithms, the particle swarm does not use selection and each solution in the population survives from the start to end of the algorithm [105]. Like a GA, a population is initialized with a number of random solutions, but each of these solutions is also imparted with a vector of randomized velocity and direction through the multidimensional search space, called a hyperspace. As the particles move through the space over time, the objective function of each solution is continuously evaluated and the more nearly optimal solutions are retained as potential solutions to the problem *(see Algorithm 2.5)* [104][117].

2.3 Evolutionary Computation Theory

Compared to traditional search methods, genetic algorithms are sometimes described as a "global search method that does not rely on gradient information." Unlike many other approaches in the field of artificial intelligence, Evolutionary Computation (EC) algorithms are considered "weak" methods because they do not heavily depend on detailed knowledge of the problem domain. Weak methods are considered beneficial in that they can be applied to solve a wider range of problems than more specialized approaches [113].

2.3.1 Schemata and the Building Blocks Theory

Genetic Algorithms (GAs) require a population of candidate solutions, and these individuals typically represent their potential solution to the problem space being searched as an array or string of values. In the case of a canonical Genetic Algorithm, the string is binary with each bit representing a gene with only two possible alleles. Less traditional GAs and other Evolutionary Algorithms may use an array or string of either real values or non-binary genes with an arbitrary number of alleles.

Regardless of the representation, these strings of values comprise the individual's genome. For a genome with l genes, we of course have a string of length l. Within this string, there are a large number of substrings of varying length, each representing a portion of the genetic code of the individual. If we consider these pieces of genetic code, it might be possible to find certain pieces that are common among individuals with high fitness and, likewise, pieces that are common in individuals with low fitness. In theory, if we could identify a sufficient number of good pieces of genetic code, we could begin to form a picture of what a highly fit candidate solution would look like.

This concept is central to the "building blocks" concept of Genetic Algorithms — that desirable solutions are made up of good building blocks, each of which tends to contribute to higher fitness in individuals in which they are present [126]. Holland's concept of schemas, or schemata, is based on this notion of desirable building blocks [92][91].

A schema, as originally proposed, is a template comprised of ones, zeros, and a "don't care" value, typically represented as an asterisk, that describes a set of all possible bit strings that can be created that conform to the template. For example, a schema of the form `1 * * 0 1` would represent all possible strings that contain the value one in the first position, the value zero in the fourth position, and the value one in the fifth position. The values contained in the second and third position can be either zero or one. In this example, this schema represents the following set of possible strings: `1 0 0 0 1`, `1 0 1 0 1`, `1 1 0 0 1`, and `1 1 1 0 1` [81][126].

All schemata are different, and vary in both length and content. Some schema, with fewer "don't care" values, can be more specific than others, while those with many "don't care" values can be more general. In addition to the values of individual bits, or genes, in the string, schemata have two important characteristics: an *order*, and a *defining length*. The order of a schema H, which is written as $o(H)$, is the number of fixed values (as opposed to "don't care" values) in the schema. The defining length of a schema, written as $\delta(H)$, is the distance between the first and last specific value position. The defining length is not the same as, and should not be confused with, the overall length of the schema. In the schema `1 0 0 * 1 * *`, the order of the schema is four, the overall length is seven, but the defining length is five. The schema `1 * * * * * *`, for instance, has both an order and a defining length of only one [81][91][126].

There are a large number of possible substrings for any given string or binary genome. Given a specific individual genome of length l, we can calculate the number of possible substrings of that single genome alone as $n = \frac{l \times (l+1)}{2}$. For a genome of length l, there are 2^l possible combinations of genes for that genome, if it is binary. If there are more than two alleles per gene, the number of possible combinations is k^l, where k is the number of possible alleles for each gene. The number of schemata to represent all of the possible substrings for a genome of length l is 3^l if the genome is binary or $(k+1)^l$ if there are more than two alleles [126][92][81].

If each individual's binary string can contain up to 3^l schemata, then in a population of n individuals, there are at most $n \times 3^l$ unique schemata in the population, or $n(k+1)^l$ in the case of non-binary strings [81]. This represents a very large amount of information available to the genetic or evolutionary algorithm. As the genetic or evolutionary algorithm progresses, the operation of selection will tend to choose the more highly fit individuals as parents. Thus, over time, the more highly fit individuals, and their presumably highly fit schemata, will tend to increase in number in the population. In essence, above-average schemata grow and below-average schemata die [81, 30].

But while selection and reproduction tend to increase the number of above-average schemata, those operations alone only work with the genetic material, or the schemata, that were present in the initial population. Without other operations to broaden the search beyond the genetic information that was already present, the selection would merely eliminate below-average individuals and multiply above-average ones until the population becomes comprised of many copies of a single individual. The operations of crossover, in which genetic material from two parents is exchanged, and mutation, in which individual genes are randomly altered, are the two primary means for injecting both new genes and new genetic substrings, and therefore new schemata, into a population. But crossover, which splits a genome into two parts, one of which is swapped with a second parent, can destroy an instance of a highly desirable schema if the location of the split in the genome occurs within the bounds of the defining length of that schema. Similarly, mutation can disrupt a desirable schema if the location of a randomly changed bit occurs at a bit location within the schema that is not a "don't care" state. So, while selection tends to increase or decrease the presence of a particular schema based on its relative fitness in the population, both crossover and mutation can decrease the presence of a particular schema by disrupting individual instances of it.

This building block hypothesis, represented by schemata, helps explain how genetic algorithms perform. Generally, shorter schemata are less likely to be disrupted by crossover than longer schemata. Over time, schemata that have a short defining length, a low order, and are highly fit tend to be selected and recombined to form new strings of potentially higher fitness, thus increasing the number of instances of that schema in the population [81, 32]. As Goldberg writes, "instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings" [81, 41].

2.3.2 The Schema Theorem

This observation gives rise to the Fundamental Theorem of Genetic Algorithms, also called the Schema Theorem: Short, low-order schemata with above-average fitness increase exponentially in frequency in successive generations [91][81, 33].

Holland's classic Schema Theorem has been expressed using different notation and nomenclature by multiple authors, but a generally accepted form can be expressed relatively clearly [6][92][126].

Given a particular schema H where $o(H)$ is the order of the schema and $\delta(H)$ is its defining length, $m(H,t)$ is the number of strings in the population belonging to schema H at time, or generation, t. Let $\hat{U}(H,t)$ be the fitness value of H at time t, or more precisely the average of the fitnesses of all instances of H in the population at time t. We can compute the expected number of instances of H in the next generation, or at time $t+1$, as $E(m(H,t+1)) = \dfrac{\hat{U}(H,t)}{\bar{f}(t)} m(H,t)$.

But we must must take into account the probabilities that H might be destroyed by either crossover or mutation. We let p_c be the probability of crossover occurring and, based on the length l of the genetic string and the defining length of the schema, set a lower bound, $S_c(H)$, for the survival of H after crossover as $S_c(H) = 1 - p_c\left(\frac{\delta(H)}{l-1}\right)$. With p_m representing the probability that any given gene will be mutated, the probability, $S_m(H)$, that schema H survives after the mutation is $1 - p_m$ for a single gene, but given a schema of order $o(H)$ then becomes $S_m(H) = (1 - p_m)^{o(H)}$. Thus, the total chance of survival, $S(H)$, of the schema by either crossover or mutation becomes $S(H) = 1 - p_c\left(\frac{\delta(H)}{l-1}\right)[(1 - p_m)^{o(H)}]$ [126].

This produces a Schema Theorem of the form: [81][92][126]

$$E(m(H,t+1)) \geq \frac{\hat{U}(H,t)}{\bar{f}(t)} m(H,t)\left(1 - p_c\frac{\delta(H)}{l-1}\right)[(1 - p_m)^{o(H)}] \tag{2.2}$$

The Schema Theorem is an inequality because of the small, but non-negligible, possibility that random mutation or crossover can create a new copy of the schema [6]. It represents a lower bound on the number of instances of the schema after the potentially destructive effects of mutation and crossover are taken into account [126, 23].

The Schema Theorem aims to describe the propagation of components of solutions in a population [68, 12] The Schema Theorem is often cited — erroneously, according to some — as an explanation for how and why a genetic algorithm is able to solve problems. Fogel writes that the conventional wisdom, including the Schema Theorem, has been shown to be "incomplete or incorrect" and that the foundations of evolutionary computation have been reconsidered [68, 12]. Many researchers raise concern with the Schema Theorem's limitations, including its inability to reconcile GA performance against stochastic search, its omission of the effects of recombination, its provision of only a lower bound for the expected number of schemata, and other issues [6][63][62][140].

Altenberg writes that the mistake commonly made with the Schema Theorem is to conclude that the growth of desirable schemata over time is related to or proof of the quality of the search being conducted by the algorithm. Indeed, Altenberg notes that the Schema

Theorem holds true in purely stochastic cases in which a GA performs no better than a random search, or in a "needle in a haystack" case in which there is one highly-fit solution while all other solutions are equally poor [6].

2.3.3 Price's Covariance and Selection Theorem

Altenberg notes that "it is the quality of the search that must be used to characterize a Genetic Algorithm," and suggests comparing a GA's ability to create new, highly fit individuals with the rate at which the same are produced by stochastic search [6].

He proposes that the Schema Theorem is an extension of Price's Covariance and Selection Theorem on population genetics. The raison d'être of a genetic algorithm is that by selecting highly fit parents and recombining them, the offspring are likely to reside in the same general "neighborhood" as the parents and therefore, also be highly fit. Altenberg maintains that this power of genetic algorithms is best seen through the covariance of the fitness of the parents and the fitness of the offspring of those parents [6, 7]. That is, when highly fit parents containing highly desirable schemata produce offspring that are also highly fit, then there is positive covariance [142].

At its simplest, Price's Theorem associates the change in frequency of a gene or genes in a population from generation to generation to the covariance between its frequency in the original population and the number of offspring produced by individuals in that population: $\Delta Q = \dfrac{\mathrm{Cov}(z,q)}{\bar{z}}$, where Q is the frequency of a specific gene or sequence of genes, $\mathrm{Cov}(z,q)$ is the covariance of the frequency of the gene in the initial population and the number of offspring produced, and $\bar{z}$ is the mean number of children produced [114, 28].

A more complete, but still simplified version of Price's Theorem can be written as $\Delta Q = \dfrac{\mathrm{Cov}(z,q)}{\bar{z}} + \dfrac{\Sigma z_i \Delta q_i}{N\bar{z}}$, where N is the size of the initial population, z_i is the number of offspring produced by individual i, and q_i is the number of copies of the gene or gene sequence in individual i [114, 30].

Langdon writes that Price's Theorem holds "for a single gene or for any linear combination of genes at any number of loci," and also accounts for other variables: sexual or asexual reproduction; random or non-random mating; diploid, haploid, or polyploid species; and species with more than two sexes [141][114, 29]. Altenberg explicitly shows its application to Genetic Algorithms [5].

2.4 Characteristics of a Genetic Algorithm

The original term "Genetic Algorithm" refers originally to the model introduced and further improved by John Holland and his students in the mid 1970s. A wider interpretation can be said to encompass any model that employs a population and selection and recombination operators to create new representations of the search space [179].

In order to solve a problem, a canonical Genetic Algorithm is comprised of five distinct components: [55]

- a binary-encoded, chromosomal, representation of solutions to the problem;
- a means to create an initial set, or population, of potential solutions;
- a evaluation function that allows the comparison and ranking of the quality of one solution to another;
- genetic operators that alter the chromosome of child solutions during reproduction; and
- parameters that further define the genetic algorithm, such as population size, mutation and crossover rates, etc.

Rowe defines the concept of a "simple Genetic Algorithm" as consisting of a population of candidate solutions, of a given population size, each of which is comprised of strings of equal length that encode a discrete representation of the problem space [150]. This representation is almost always represented as a vector or array of natural numbers of length

n such that $\vec{x} \in \mathbb{N}^n$, when the the traits to be represented are either binary or comprise a finite set of alleles. In cases where the traits are not represented by discrete values, the vector or array of real numbers is described as $\vec{x} \in \mathbb{R}^n$ [85].

Rowe defines a "generation" as the state of the population at any given time-step during the iterative operation of the algorithm [150].

The process of reproduction (i.e. producing a new candidate solution, called the "child") is usually accomplished by mating and consists of choosing two or more candidate solutions, called the "parents", from the general population and then combining portions of each of the parents in such a way to produce a new, distinct candidate solution. Although two-parent reproduction is the standard in a canonical Genetic Algorithm, and still remains most common, single-parent reproduction and multi-parent reproduction ($\rho \gg 2$) are also possible.

A traditional process of reproduction on a population of size n is repeated exactly n times, resulting in a set of entirely new individuals in each subsequent generation and preserving none of the individuals from the parental generation. Although this is a commonly accepted approach, it can result in the loss of a superior candidate solution whose full genome does not survive into subsequent generations [81][126]. There are many variations on basic reproduction to counter this, including: cloning; probabilistic crossover; elitism; and steady-state reproduction strategies that gradually improve a population rather than completely replacing it each generation.

Rowe elaborates that the basic process of mating is typically built upon three distinct "genetic operators" that act upon or create new individuals in the population: selection, crossover, and mutation [151]. Several aspects of a genetic algorithm can be generalized and reused from one problem domain to another. However, two main components — the problem encoding and the evaluation function — are problem dependent and not easily generalizable [179].

2.4.1 Fitness

Of critical importance to evolutionary algorithms is the ability to determine the desirability of one potential solution compared to another. With a pool of many potential solutions, such algorithms need the ability to rank solutions relative to each other, chose parents from among the more desirable solutions, and recombine the genomes of those parents to produce — hopefully — more desirable child solutions.

To determine the relative quality or acceptability of an individual in the population, a computable quantity must be defined as a basis for comparison. The exact criteria that are used to rank solutions relative to each other, although typically computed on a numerical scale, is irrelevant as long as the ranking can occur. Fogel writes "within evolutionary algorithms, any definable payoff function can be used to judge the appropriateness of alternative behaviors. There is no restriction that the criteria be differentiable, smooth, or continuous" [63, 5]. Fogel and Goldberg call these ranking measures "payoff values," but the term is now more commonly referred to as a "fitness value" [81].

Fogel relates fitness in evolutionary algorithms to the concept in biological evolution of the ability to survive and reproduce in a specific environment [66]. This concept translates directly to software simulation of evolution in Genetic Algorithms. Individuals have a fitness quantity associated with them that is designed to indicate the acceptability of the individual as a potential solution to the problem (i.e. environment) being explored. The fitness value is also used in judging the acceptability of an individual as a parent during selection in the mating process and in other areas as well.

Fitness can also be used to assess the overall quality of a population, as the population of individuals can be considered to have evolved if the average fitness of the population has increased. Since the intent and result of the genetic algorithm is to produce more fit individuals through genetic recombination, with fitness as the measurable quantity, it is easy to envision how a genetic algorithm accomplishes evolution by raising fitness levels over time [63][24].

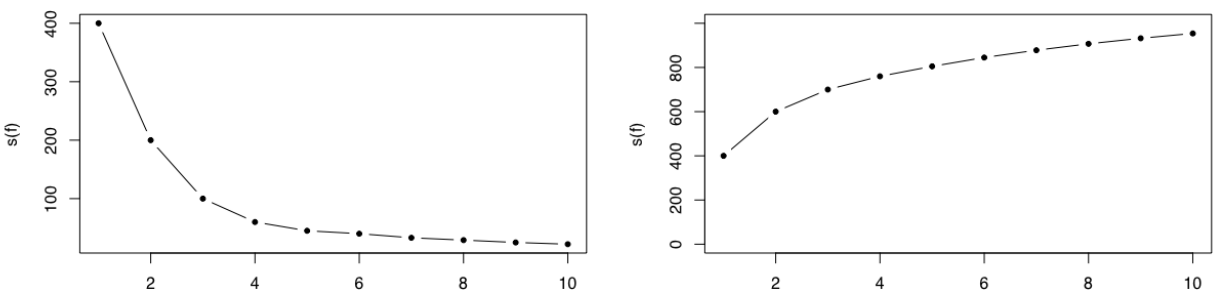

Figure 2.2: **Fitness Distribution and Cumulative Fitness Distribution.** A typical fitness distribution curve for a population (left) and a cumulative fitness distribution curve for the same population (right).

Given a population with μ individuals, the fitness distribution of all or part of the population can be ordered such that $f_1 < f_2 < ... < f_{n-1} < f_n$ where $n \leq \mu$. In many Genetic Algorithms, a population is typically comprised of a few highly fit individuals and a majority of less fit individuals, and this distribution can be charted. The cumulative fitness distribution can be used as a measure of what portion of a population is less than a certain fitness value [23]. A typical fitness distribution curve and corresponding cumulative fitness distribution curve are shown in Figure 2.2.

Developing a feasible fitness function for any given problem is, perhaps, the biggest challenge in correctly implementing evolutionary and Genetic Algorithms. Construction of the fitness function is critical in Evolutionary Computation because "(i)nappropriate descriptions of the performance index lead to generating the right answer to the wrong problem" [68, 7]. Knowledge of the domain being modeled is essential to devising a reasonable fitness function for real-world problems, Fogel writes: "Evolutionary algorithms offer a framework such that it is comparably easy to incorporate such knowledge... Incorporating such information focuses on the evolutionary search, yielding a more efficient exploration of the state space of possible solutions" [63, 5].

However, the simplicity of the concept of a fitness function belies the difficulty, when dealing with very complex evolutionary problems, of reducing a multivariate candidate solution to a single value that can be used for, assuming no two solutions can be equal, a binary decision on inequality.

2.5 Evolution Strategies

Hansen et al. define an evolutionary strategy as "an iterative (generational) procedure" where in "each generation, new individuals (offspring) are created from existing individuals (parents)" [85, 874]. Various symbols, abbreviations, and notation methods have been created to define different evolution strategies, but here we will use the nomenclature favored by Beyer and Schwefel, Hansen et al. and others [22][84]. Notation for evolutionary strategies and selection schemes is generally expressed here in the form $(\mu/\rho \overset{+}{,} \lambda)$, which defines such schemes in terms of the following symbols, which we will use henceforth: [22][84][167]

$P(t)$, a set of individuals (i.e. the population) at generation t

$\vec{a} \in I$, a single individual in the space of individuals I

$\mu \in \mathbb{N}$, the number of potential parents (i.e. the population size)

$\rho \in \mathbb{N}, \rho \leq \mu$, the number of parents selected for recombination

$\lambda \in \mathbb{N}$, the number of offspring (i.e. the offspring population size)

2.5.1 General Evolution Strategies

An evolution strategy is an implementation of one of various recombination techniques, which are the act of combining information from multiple parents to generate a new offspring. Although *multi-recombination*, which involves producing a single offspring from more than two parents ($\rho > 2$), is not unusual, in the context of Genetic Algorithms recombination is most commonly accomplished with exactly two parents ($\rho = 2$) producing a single offspring [84, 7].

There are several recombination operators, or methods, possible to accomplish the act of combining genetic information from multiple parents, and the nomenclature of each method varies among authors. Tettamanzi and Tomassini classify these generally as either "discrete" or "intermediate" recombination strategies [167, 24]. Hansen et al. also define these two categories, but add a third category called "weighted" recombination [84, 7].

Discrete. Discrete recombination, sometimes called "dominant recombination" generally and as "uniform crossover" in the context of genetic algorithms, are those techniques in which each atomic part of a child's genome is copied exactly from one of the available parents, often chosen stochastically. Given a genome that consists of l binary genes, discrete recombination, in the simplest terms, can be envisioned as the creation of a child by iterating through each locus in the genome and, for each position l, randomly selecting the lth gene from one of the two or more available parents and copying it exactly to the child.

Intermediate. In intermediate recombination, each part of the child's genome is a linear combination of the corresponding parts from each of the parent individuals' genomes.

Intermediate recombination can be envisioned — again, in an extremely simple example — as a genome of l real values in the range 0..1 such that a child is created by iterating through the each locus l such that the value of an individual gene $k_l = \dfrac{(p_{1_l} + p_{2_l} + ... + p_{n_l})}{n}$, or $k_l = \dfrac{\sum_1^n p_{n_l}}{n}$, where p_n is one of n parents and p_{n_l} is the value of the gene at locus l of that parent. In the case of recombination from a pair of parents, this is more simply expressed as $k_l = \dfrac{p_{1_l} + p_{2_l}}{2}$.

Weighted. Weighted recombination is a generalization of intermediate recombination in which, instead of a simple average of all ρ parents, a weighted average is used. All parents are ordered such that inferior parents are always weighted less than better parents and the weight given to the gene of a specific parent is adjusted based on the ordering. In this way, the genes of stronger parents exert more force in the recombination than the genes of weaker parents.

Regardless of the means of recombining individual genes, recombination strategies vary greatly depending on several factors: whether a single parent or multiple parents are required; the method employed to select parents; the number of parents selected to form a breeding pool; the number of children produced by each parent or parents; and the number of children that are allowed to enter the subsequent generation.

Cloning

Among the simplest approaches for producing an offspring involves *cloning*, the process of selecting a single, usually highly fit, parent and producing an exact copy of that parent. Cloning, although simple, has particular value in recombination schemes that require the complete replacement of the pool of candidate solutions with a new set of child solutions in each generation. In such schemes, parents can only contribute part of their genetic material to future generations through recombination in the form of a child. The parent's entire genome cannot be passed in its entirety and, thus, an exceptionally fit candidate solution is potentially lost. Cloning provides a means for a parent to create an identical child and, therefore, allows a desirable genome to propagate without alteration into subsequent generations. Cloning may be an distinct operation or it may occur as a result of probabilistic crossover. "Pure crossover" is described as obtaining an identical individual, or clone, when crossing an individual with itself [151].

Cloning may be used as a supplement to other recombination strategies, or it may be used as the sole recombination strategy. When used as the only method for producing child solutions, an additional means for injecting new genetic information into the population is necessary, otherwise each subsequent generation would be essentially identical to the previous. *Mutation* is the method most commonly coupled with cloning to provide a means for new traits to enter the population *(see Section 2.6.4)*. In such a situation, the parents selected for reproduction are cloned, and a mutation operator is applied that might mutate, or change, a gene to a different allele, usually randomly and with a fairly low probability of mutation occurring for any given gene.

Orgy

On the opposite side of the spectrum from cloning, in which a single parent produces an identical or near-identical offspring, is the concept of *orgy*, in which all individuals in the population are combined simultaneously to produce a new offspring [167, 24]. In an orgy, the child's gene for any given locus is chosen, usually at random, from among all the alleles available among all individuals in the population. The probability that an allele is selected from a specific parent may be based on an equal distribution, may be based on the frequency of the allele's occurrence, or it may be weighted in some manner based on the desirability (i.e. fitness) of a specific parent.

2.5.2 Selection and Recombination Schemes

Generational versus Steady-state Reproduction

Central to selection and recombination strategies are the number and method of selecting parents for reproduction, the number of offspring produced per parent, and how many children are allow to persist into the subsequent generation. Therefore, many evolutionary schemes can be defined in terms of the number of parents in each generation and the number of children produced prior to each new generation. This is most commonly expressed with μ, also often referred to as simply n, representing the number of parents in the population and λ, or m, denoting the number of children within one generation. Other authors, such as Schwefel, use different symbols than described here [159].

Generational approaches to reproduction select parents from the population, recombine or mate those parents to produce a number of offspring equal to the size of the population, and then replace the entire population with the set of newly created children. In such approaches, the entire population live and dies in a single generation and is replaced by an entirely new, but genetically similar, population [150].

In steady-state approaches to reproduction, once the search algorithm finds a relatively good candidate solution and places it into the population, that individual remains in the population until replaced by a better one. Whitley observes that "this means that the algorithm is less prone to 'wander' in the search space and will maintain an emphasis on the best schemata found so far" [178]. Steady-state reproduction also removes the need for probabilistic crossover, which is a technique used to allow parents to produce identical children and, therefore, sometimes propagate their genome into subsequent generations [178].

Yet other approaches blur the lines between steady-state and generational approaches and allow a significant portion of parents to remain while discarding many less-fit children to keep the population size constant.

Selection and Recombination Scheme Notation

A general notation for expressing selection and recombination schemes in evolutionary strategies is in the form $(\mu/\rho\overset{+}{,}\lambda)$, where μ, ρ, and λ are as described above and the $\overset{+}{,}$ symbol signifies either of two general methods for determining which individuals are allowed to enter subsequent generations. In the case of the comma symbol, the entire population is replaced by the new offspring in each subsequent generation, and selection is applied to the offspring to determine which are allowed to survive [36]. In the plus notation, parents do not automatically die out, or "age," and the μ best in the combined set of all parents, P, and offspring, λ, are chosen, or selected, to survive and become the next generation. The details of exactly how many parents survive and how many children live can vary, but most "plus" schemes employ some form of elitism, whether strict or loose, which seeks to guarantee a monotonically increasing population fitness [14][84].

A subscript may be added to ρ to indicate the type of recombination, such as ρ_I or ρ_W for intermediate or weighted recombination [84]. In casual notation, ρ is more often omitted entirely and recombination notation is simplified to the form $(\mu \overset{+}{,} \lambda)$. Sometimes the notation is expanded to include the value κ, which represents the maximum age permitted for any individual, and the strategy's notation becomes (μ, κ, λ), where a pure 'plus' scheme is equivalent to $\kappa = \infty$ and a pure 'comma' scheme is equivalent to $\kappa = 1$ [84].

A description of several general evolution strategies follows:

$(1 + 1)$-EC and $(1 \overset{+}{,} \lambda)$-EC. The $(1 + 1)$ strategy is an extraordinarily simple one in which the population may consist of only a single individual that is cloned, usually with a mutation operator being applied to the child. Elitism is employed, either between the parent and mutant child or amongst the set of all parents and mutants combined. In $(1 + 1)$-EC, or its multiple-child variant $(1 + \lambda)$-EC, the mutant(s) are compared to the parent and the single best of the these is retained. In the $(1, \lambda)$-EC, the μ most fit of all parents and mutants are allowed to enter the next generation while the remainder perish.

(μ, λ)-EC. Another relatively simple scheme in which a number of children, λ, are produced from a set of parents. Traditionally, the terms "population" and μ are considered interchangeable, but it is possible for different implementations to designate a separate "parental pool" that may not include all individuals in the population or may contain multiple copies of some individuals.

The number of children may equal or exceed the number of individuals in the population, such that $\mu \leq \lambda < \infty$, but the μ most fit of the children are chosen to enter the new generation. The entire population perishes after every generation and an entirely new, but presumably genetically similar, population of children replaces it [84][159].

$(\mu + \lambda)$-EC. This is a minor variation on (μ, λ)-EC that allows parents the opportunity to persist from generation to generation. In this scheme, instead of just the children, the combined set of all parents and offspring, $(\mu \cup \lambda)$ or more simply $(\mu + \lambda)$, become the source individuals to be included in the next generation. The μ strongest individuals, based on fitness, are chosen, permitting strong parents to survive but allowing weaker parents to make room for more-fit children [28][84][159].

2.6 Genetic Operators

2.6.1 Basic Nomenclature

Early researchers identified four core operations that, when performed repeatedly and in various combinations, allow evolutionary and Genetic Algorithms to converge toward more optimal solutions. Holland first defined the three operations of *simple crossover*, *simple inversion*, and *mutation* [92, 121]. The additional operation of *selection*, while not explicitly listed as an independent operation by Holland, is fundamental to the function of a Genetic Algorithm. The term crossover is sometimes considered synonymous with the term *recombination*, and is often used in literature in place of it, but recombination is more broad and includes additional methods of genetic transfer than offered by crossover alone. Inversion, although one of the operators originally proposed by Holland, is seldom seen in GAs today.

2.6.2 Selection

Selection is the process by which individuals in the population are chosen to become parents and, through mating, produce a child that survives into the subsequent generation. Selection is often considered the central concept to evolution in that as new characteristics are produced through mating their value can only be assessed through competitive selection and, if not found wanting, those new characteristics can enter and spread through the population. Fogel summarizes selection's role by paraphrasing Darwin: "Useful variations have the best chance of being preserved in the struggle for life, leading to a process of continual improvement” [64][53, 130].

Blickle and Thiele note that many selection methods are probabilistic in nature and make use of the inherent distribution of fitness values in a population. They define a selection operator Ω “as a function that transforms a fitness distribution into another fitness distribution” s' such that $s' = \Omega(s, pars)$, where $pars$ is an optional set of parameters to the selection function [23, 7].

Fogel explicitly explains selection as a means to increase the fitness of a population [66]. There are a large number of accepted methods for selection, the most common of which include Binary Tournament Selection (BTS) and Fitness Proportionate Selection (FPS).

In a general selection operation, a pair of the — ideally — more-desirable instances in the population is selected for recombination. They are then mated to produce a new instance, the “offspring” or “child.” It is hoped that the child will have a higher fitness than at least some of the individuals already in the population. Depending on the evolution strategy, the child may automatically enter the next generation or, if it is strong enough, the child can enter the population and the least fit member of the population dies off to make room for it.

There are two general categories of selection schemes for mating in genetic algorithms: *fitness-independent* selection and *fitness-based* selection.

Fitness-independent. In a fitness-independent scheme, the fitness of individuals in the population does not affect their likelihood of selection for mating nor the transfer of their genome during recombination [84]. In such schemes, some other method or methods are needed to ensure more-fit individuals persist and propagate their genetic material, so that over time the overall fitness of the population increases [85].

Fitness-Based. In a fitness-based selection scheme, individuals in the population are in some way ranked according to their relative fitness, and the choosing of parents is done in a

Algorithm 2.6: A simplified tournament selection algorithm with tournament size t, where $1 \leq t \leq \mu$. [23, 14]

```
Data: The population $P(\tau)$ and tournament size $k \in \{1, 2, ..., m\}$
Result: The population after selection $P(\tau)'$
1 tournament(k, $J_1, ..., J_n$):
2 begin
3     for $i \leftarrow 1$ to $N$ do
4         $J_i' \leftarrow$ best fit individual out of $k$ randomly picked individuals from $\{J_1, ..., J_n\}$
5     end
6     return $\{J_1', ..., J_n'\}$
7 end
```

proportional or otherwise biased manner such that more-fit individuals tend to be selected more often. In such schemes, it is usually through selection alone that more-fit solutions tend to propagate into subsequent generations and that the overall fitness of the population increases over time [84].

Many experts, including Rowe, favor fitness-based, proportional or ranked, selection methods: "We usually require a selection operator to assign a higher probability to elements that have a higher fitness" [151, 921].

There are many differing selection algorithms that have been proposed. These can be divided into several categories, the first of which include weighted, or fitness-based, selection methods such as: Tournament Selection, or Binary Tournament Selection (BTS); Truncated Selection; Roulette Wheel Selection (RWS) or Fitness Proportionate Selection (FPS); Linear Ranking Selection; Exponential Ranking Selection; and Elitist Exponential Ranking Selection. A second category is comprised of fitness-independent methods and includes Stochastic Universal Sampling. Finally, a third category of hybrid methods includes Roulette Tournament Selection (RTS) and Queen Bee Selection.

Tournament Selection

Although there are many variants, Tournament Selection is considered almost universally to be the simplest of selection algorithms, both in ease of understanding and in implementation. A specific number, k, of individuals are chosen stochastically from the population and from these the most-fit individual is selected and copied into the the pool *(see Algorithm 2.6)*. This is repeated m times, with m representing the desired number of parents (usually two). The value k is termed the "tournament size," and tournament selection schemes are generally described a "$k = n$ tournament selection," where n is the number of competing individuals in each tournament. When tournaments are held between two individuals, $k = 2$, it is called a "binary tournament" and this specific method is referred to as Binary Tournament Selection (BTS).

A consequence of tournament selection, which can be mathematically derived, is that individuals with the lowest fitness values have a reproduction rate of almost zero which leads to a phenomenon known as "loss of diversity" [23, 19]. As the tournament size increases, a certain proportion of the least-fit individuals in the population are lost from one generation to the next. The proportion of loss is not linear with respect to the tournament size. That is, the number of individuals lost increases greatly with the size of the tournament. As much as half of the population can be lost with a tournament size of only 5 [23, 19].

Truncated Selection

Truncation selection is a method in which, given an ordered population and a truncation threshold T, only the T best individuals can be selected. Those individuals that fall beneath the truncation threshold have a probability of selection of zero and the individuals above the threshold all have an equal probability of selection. Put simply, truncated selection forms a pool of potential parents based on individual fitness such that an arbitrary number or arbitrary percentage of the more-fit candidate solutions are allowed mate and the remainder

Algorithm 2.7: A simplified truncated selection algorithm. [23, 23]

```
Data: The population $P(\tau)$ and truncation threshold $T \in [0, 1]$
Result: The population after selection $P(\tau)'$
1 truncation($T$, $J_1, ..., J_n$):
2 begin
3     $\bar{J} \leftarrow$ population $J$, sorted by increasing fitness
4     for $i \leftarrow 1$ to $N$ do
5         $r \leftarrow$ random$\{[(1-T)N], ..., N\}$
6         $J'_i \leftarrow \bar{J}_r$
7     end
8     return $\{J'_1, ..., J'_n\}$
9 end
```

are not *(see Algorithm 2.7)*. Given all available solutions, the n highest ranked individuals are chosen to form the parental pool [84][52].

Truncated selection is directly related to selective breeding techniques in animal populations [33][52], and was introduced as the Breeder Genetic Algorithm (BGA) by Mühlenbein and Schlierkamp-Voosen in 1993 [130].

Because a significant percentage of the population is eliminated from potential selection, in order to keep the population size constant, this has the effect of increasing the selection chances of the remaining individuals by a multiplication factor of $\frac{1}{T}$, and there is subsequently a high loss of diversity $p_d(T) = 1 - T$.

Roulette Wheel Selection

Roulette Wheel Selection (RWS), also known as Fitness Proportionate Selection (FPS), is considered the first selection method proposed for use in genetic algorithms [90][92]. In a proportional scheme, parents are selected based on a probability value that varies depending on the relative fitness of one individual to another, such that more fit individuals tend to get selected more often. This method depends on non-negative fitness values and the probability of selection of an individual is proportionate to its fitness value: $p_i = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$,

Algorithm 2.8: A roulette wheel (fitness proportionate) selection algorithm [23, 40]. (Reverse bracket notation indicates excluded endpoint(s) for the given interval.)

```
   Data: The population $P(\tau)$
   Result: The population after selection $P(\tau)'$
1  proportionate($J_1, ..., J_n$):
2  begin
3      $s_0 \leftarrow 0$
4      for $i \leftarrow 1$ to $N$ do
5          $s_i \leftarrow s_{i-1} + \frac{f_i}{M}$
6      end
7      for $i \leftarrow 1$ to $N$ do
8          $r \leftarrow \text{random}[0, s_N[$
9          $J_i' \leftarrow \bar{J}_l$ such that $s_{l-1} \leq r < s_l$
10     end
11     return $\{J_1', ..., J_n'\}$
12 end
```

which is sometimes expressed as $p_i = \dfrac{f_i}{\mu M}$, where M is the average fitness of the population [23, 40][84].

Roulette Wheel Selection, because of the nature of the algorithm, is considered most suitable for maximization problems and unsuitable for minimization problems. However, some minimization problems, especially those with a known upper bound, may still employ RWS by simply using a modified fitness value such as $bound_{upper} - f(x)$ to reverse the slope of the curve.

A simplified implementation of the RWS algorithm is accomplished by first calculating the sum of the fitness values of all individuals in the population, $S = \sum_{1}^{\mu} f(x)$. A random value r is then generated in the range $0...S$. Finally, iterate through the population, which is sorted by either increasing or decreasing fitness, examining the fitness value of each individual and adding that to a running sum of the fitness values seen so far. When the running sum exceeds r, then the current individual is selected *(see Algorithm 2.8)* [155].

The slope of the fitness distribution curve in Roulette Wheel Selection may be either steep or shallow, depending on how closely or how widely the fitness values of the individuals in the population vary from one another. The presence of a "super individual" with fitness vastly higher than the rest of the population can dominate proportionate selection [15]. When all individuals in the population have nearly identical fitness values, there is little difference between RWS and a completely stochastic selection method.

The scaling of the fitness values heavily influences the rate of selection of individuals in the population. Blickle and Thiele illustrate this with the example of a population of ten individuals whose fitness values vary from 1 to a maximum of 11. The selection probability for the best individual can be calculated as $p_b \approx 16.6\%$ and for the worst individual $p_w \approx 1.5\%$ [23, 40]. However, taking the same population and multiplying the fitness values by a factor of 100 produces a probability of selecting the best individual with $p'_b \approx 10.4\%$ and the worst individual with $p'_w \approx 9.5\%$, which results in little difference between the selection rate of the best and worst individuals in the population [23, 41].

Mitchell calls this "pressure convergence" [126, 125] and Whitley similarly describes this phenomenon in terms of selective pressure [178]. Consider the same scenario in which a population's fitness ranges from 100 to 1100, with an average of 550. The selective pressure of the top-ranked individual is $SP = 1100/550 = 2.0$. However, as the population improves over time, imagine a later generation with fitnesses ranging from 1000 to 1200 with an average of 1100. The selective pressure of the top individual is now $SP = 1200/1100 = 1.09$, which may not be adequate to keep the search progressing [178, 2]. This situation becomes more acute as the the population becomes more homogenous or there is less variation in the fitness of individuals.

Fitness Proportionate Selection can suffer from two drawbacks: a search can stagnate if there is insufficient selective pressure, and a search can prematurely converge at an inferior solution because selection narrows the available genotypes too quickly [178, 2][126, 125].

Scaling Methods. Because of the tendency described above, proportionate selection is said to be *translation variant*, and a number of scaling methods have been proposed (e.g.

linear static scaling, linear dynamic scaling, exponential and logarithmic scaling, sigma truncation or sigma scaling) [16][23][30][81][82][122][126]. An alternative method to compensate for this is "overselection" of a certain percentage of the best individuals [23][108][109]. This type of scaling is considered essential, and not just an optimization, to proportionate selection algorithms [23][131]. Blickle and Thiele consider the abundance of undesirable properties in fitness proportionate selection and conclude that it is a very unsuitable selection scheme [23, 42].

Ranked Selection Methods

Linear Ranking Selection. For ranking selection, the population is ordered according to increasing fitness such that the probability of selection is linearly proportionate to their ranking. Whitley describes ranking succinctly: "Ranking acts as a function transformation that assigns a new fitness value to a genotype based on its performance relative to other genotypes" [178].

A typical probability function for selection is $p_i = \frac{1}{N}(\eta^- + (\eta^+ - \eta^-)\frac{i-1}{N-1}); i \in \{1, ..., N\}$ [23, 27] With this, the probability that the worst individual will be selected is $\frac{n^-}{N}$ and the probability that the best will be selected is $\frac{n^+}{N}$ *(see Algorithm 2.9).* No two items are ranked identically, and in the case of multiple individuals with identical fitness, they are arbitrarily ranked respective to each other.

For this reason, ranking selection has been suggested as a resolution for problems that can arise in proportionate selection methods, particularly early convergence caused by "super individuals" and reduced selective pressure as populations become more homogenous [40, 138][82][178].

Linear Ranking has been shown to provide a more even chance of individuals being selected, or what Razali and Geraghty describe as "constant pressure," than proportionate selection. This is especially true when the fitness values in the population vary greatly, as

Algorithm 2.9: A linear ranking selection algorithm [23, 28].
(Reverse bracket notation indicates excluded endpoint(s) for the given interval.)

```
    Data: The population $P(\tau)$ and the reproduction rate of the worst individual
          $n^- \in [0,1]$
    Result: The population after selection $P(\tau)'$
1   linear_ranking($n^-$, $J_1, ..., J_n$):
2   begin
3       $\bar{J} \leftarrow$ population $J$, sorted by increasing fitness
4       $s_0 \leftarrow 0$
5       for $i \leftarrow 1$ to $N$ do
6           $s_i \leftarrow s_{i-1} + p_i$
7       end
8       for $i \leftarrow 1$ to $N$ do
9           $r \leftarrow$ random$[0, s_N[$
10          $J'_i \leftarrow \bar{J}_l$ such that $s_{l-1} \leq r < s_l$
11      end
12      return $\{J'_1, ..., J'_n\}$
13  end
```

the effects of "super individuals" that might otherwise dominate a proportionate selection scheme are mitigated [127].

Often in rank-basked selection, a selective pressure factor can be used to scale the ranking of individuals. A typical approach is to select a selective pressure value, SP, such that $1 \leq SP \leq 2$. The ranking ($f'(x)$) of individuals is then scaled according to the following function: $f'(i) = 2 - SP + (2(SP-1)\frac{i-1}{\mu-1})$. This function results in a scaled ranking such that that the sum of the maximum rank value and the minimum rank value equal two: $f'_{max} + f'_{min} = 2$.

The selective pressure can be small (e.g. $SP = 1.1$), resulting in a maximum and minimum scaled ranking of 1.1 and 0.9 respectively, or it can be large (e.g. $SP = 2.0$), resulting in a maximum and minimum scaled ranking of 2.0 and 0.0 respectively. A selective pressure of 1.5 implies that the top ranked individual is 1.5 times more likely to reproduce than the the median individual in each generational cycle [178]. A selection pressure of 1 represents no pressure and reduces the equation to a constant, resulting in uniform random selection [40][127].

Algorithm 2.10: An exponential ranking selection algorithm. [23, 35]
(Reverse bracket notation indicates excluded endpoint(s) for the given interval.)

```
Data: The population $P(\tau)$ and the ranking base $c \in [0, 1]$
Result: The population after selection $P(\tau)'$
exponential_ranking(c, $J_1, ..., J_n$):
begin
    $\bar{J} \leftarrow$ population $J$, sorted by increasing fitness
    $s_0 \leftarrow 0$
    for $i \leftarrow 1$ to $N$ do
        $s_i \leftarrow s_{i-1} + p_i$
    end
    for $i \leftarrow 1$ to $N$ do
        $r \leftarrow \text{random}[0, s_N[$ $J_i' \leftarrow \bar{J}_l$ such that $s_{l-1} \leq r < s_l$
    end
    return $\{J_1', ..., J_n'\}$
end
```

However, the principle of selective pressure can be frustrated by the presence of duplicates of an individual in the population, in both proportionate selection and other methods. It is easy to conceive of a situation in which two identical individuals with mediocre fitness together can exert more influence than the most-fit individual by itself [178].

Once rank scaling have been accomplished, the selection can proceed as with roulette wheel selection, but with the scaled ranks in lieu of actual fitness values.

Exponential Ranking Selection. Exponential ranking is substantially similar to linear ranking with the exception that the probability of selection of an individual is exponentially weighted. The base of the exponent, $0 < c < 1$, is a parameter of the selection function such that the closer c is to 1 then the lower the growth of the exponential calculation in the method [23, 34]. Given a population ordered by increasing fitness, the probability of selection becomes $p_i = \frac{c^{N-i}}{\sum\limits_{j=1}^{N} c^{N-j}}; i \in \{1, ..., N\}$, which can be simplified to $p_i = \frac{c-1}{c^N - 1} c^{N-i}; i \in \{1, ..., N\}$ *(see Algorithm 2.10)* [23, 34].

Algorithm 2.11: A roulette tournament selection algorithm [23][84][144].
(Reverse bracket notation indicates excluded endpoint(s) for the given interval.)

```
Data: The population $P(\tau)$
Result: The population after selection $P(\tau)'$
proportionate_tournament($K_1, ..., K_n$): begin
    $s_0 \leftarrow 0$
    for $i \leftarrow 1$ to $N$ do
        $s_i \leftarrow s_{i-1} + \frac{f_i}{M}$
    end
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $t$ do
            $r \leftarrow$ random$[0, s_N[$
            $J'_t \leftarrow \bar{J}_l$ such that $s_{l-1} \leq r < s_l$
        end
        $J''_i \leftarrow$ the more fit of $\{J'_1, ..., J'_t\}$
    end
    return $\{J''_1, ..., J''_n\}$
end
```

Elitist Exponential Ranking. Selection can be accomplished in an elitist fashion using an exponential ranking scheme that vastly favors more-fit solutions without eliminating the possibility of the worst-fit solution being selected. In such a scheme, the probability of any given solution in a ranked population being selected follows a exponentially decreasing curve, such that the probability of the k^{th} individual in the ranked population will be selected at a frequency inversely proportional to 2^k, or such that $P(k) = \dfrac{1}{2^k}$ [167, 187].

Hybrid Methods

Roulette Tournament Selection. One of the more interesting hybridizations of fitness-based and fitness-independent techniques combines both the proportionate selection seen in Roulette Wheel Selection (RWS) and the randomized approach seen in Binary Tournament Selection (BTS). Roulette Wheel Selection uses a weighted selection that has a higher probability of selecting fit individuals, but there still remains a significant chance of selecting an inferior individual.

By performing RWS twice, and then employing Binary Tournament Selection (BTS) to chose the best of the two, the chances of ending up with an inferior individual as a parent are lessened. We call this two-phase selection method Roulette Tournament Selection (RTS), and it has been studied by Hansen et al., Rahman et al., and others [84][144].

Queen Bee Selection. Queen Bee selection combines both random selection and elitist (ranked) selection into a single operator. The name is derived from the concept of a queen bee, which is usually the only sexually mature female and mother to most or all of the bees in the hive [149]. The queen mates with available drones in the hive, such that all descendants come from the single queen but from one of many drones. In queen bee selection, the most-fit individual in the population, according to fitness, is always selected as one parent. The other parent is usually chosen at random. This randomly selected, second parent may be chosen for recombination with the queen for all genes during mating, or a second random parent may be chosen for each individual gene in the genome. In either case, recombination of each gene between the queen and the second parent is usually a random coin toss [11][101][102][103][144].

Reducing Population Size

After selection, recombination, and mutation are completed, many EC algorithms employ additional pruning operations to either modify the pool of child solutions before it enters the population or to cull individuals from the pool of children or the population itself. For instance, it may be desirable to take steps to ensure certain highly fit individuals survive from one generation to the next. Or it may be desirable for algorithms to prevent extremely unfit individuals from entering the population at all. Additionally, many forms of EC algorithms, whether generational or steady-state, come to a point at which there are more potential individuals in the population than the population size limit, μ, allows. This may occur when a mating and recombination method produces more than μ children each generation, or it may result from algorithms that combine children and parents into a single pool, but the algorithm must choose from among the available individuals to determine which shall survive.

Environmental Selection. Environmental selection is the general name given to the selection operation used in Evolutionary Computation algorithms that must limit the number of individuals allowed to enter subsequent generations. It is generally applied after recombination and mutation have been completed and reduces the population size to the desired population size limit, μ [42]. The criteria for environmental selection can vary and may be as simple as a fitness-based truncation selection [85], or it may be more sophisticated and choose a set of individuals with good distribution and convergence performance [42, 595]. Cheng et al. show that employing an environmental selection algorithm that combines both a directional density function and favorable convergence function provides continued selective pressure in a multi-objective optimized search problem.

Elitism Elitism is a similar operation that can be applied alone or in addition to environmental selection to force the Evolutionary Computation algorithm to retain some number of the best individuals — the "elite" members of the population — by copying them directly into the subsequent generation [57]. Otherwise such highly fit individuals could potentially be lost if they are not selected to reproduce. Similarly, important parts of highly fit individuals' genomes could be lost or corrupted during the crossover or mutation operations. Several researchers have found that elitism significantly improves the Genetic Algorithm's performance [126].

Infant Mortality Infant mortality is another related concept that attempts to prevent poor quality individuals from entering the population at all. The concept of infant mortality in Genetic Algorithms is based upon the assumption that a very unfit or weak offspring solution would not survive long in nature [3][115]. A simple implementation of infant mortality in GAs is to merely prevent a very unfit child solution from entering the subsequent generation based on a fitness limit. This threshold may be a fixed, proportionate, or variable value. To keep population levels consistent, extra children may be produced from the pool of available parents through normal selection and recombination, or a highly fit parent may be granted longevity and allowed to survive into the subsequent generation.

Overaging. Overaging is an approach to limit the longevity of highly fit individuals that may persist from generation to generation for Evolutionary Computation algorithms that allow parents to survive into subsequent generations. With overaging, individuals that exceed a certain maximum age, κ where $1 \leq \kappa \leq \infty$, are removed from the population. Overaging can be useful to slow the takeover of the population by one or more highly fit individuals. Evolutionary Strategies (ES) are an example of algorithms that employ the extremes of overaging, with $(\mu/\rho, \lambda)$ strategies representing overaging with $\kappa = 1$ and $(\mu/\rho+\lambda)$ strategies representing overaging with $\kappa = \infty$ [85]. Overaging can also be addressed as a step in an environmental selection algorithm [84, 5].

2.6.3 Recombination and Crossover

More than any other, the *recombination* operator exerts great influence on the performance of a genetic algorithm in terms of speed and the quality of the solutions returned. Recombination forms half of the "exploration versus exploitation" dynamic, with the other half being mutation, that makes genetic algorithms succeed. "Mutation serves to create random diversity in the population," writes Spears, "while (recombination) serves as an accelerator that promotes emergent behavior from components" [163]. That is, while mutation, discussed below, can add new genetic material to a population, recombination attempts to exploit the existing genetic material to create higher quality combinations of genes. The distinction between recombination and *crossover* is subtle, and Muhlenbein et al. explain that recombination refers to the mixing of variables, or genes, between parents and crossover is the mixing of the values of a variable, or gene [129].

Crossover is a specific, and the most common, method of accomplishing recombination and involves copying a number of genes from each of two or more parents to produce a chromosome representing a child.

The child receives genetic material from two or more parents in some proportion, which is usually an equal chance for all parents. In nature the percentage of genetic material contributed by each parent is sometimes referred to as the crossover rate. In mammals, this is sometimes as much as 30 percent higher for females than males [66]. In Evolutionary Computation (EC) algorithms, many approaches to crossover have been suggested, but the two simplest — and most common — approaches are single-point crossover and uniform crossover [17].

Single Point Crossover. One of the simplest implementations of crossover — both conceptually and programmatically — is single point crossover, in which a locus, or position, along the genome is randomly selected in the range of $(1..n-1)$ as a crossover point. During recombination, a simple copying of the portion of the genome up to and including the crossover point from the first parent and then the remaining portion from the second parent is all that is needed *(see Algorithm 2.12)*.

Single-point and other crossover methods can be implemented to produce one child at a time or to produce a pair by repeating the algorithm with an inversion of the ordering of the parents to produce two children such that the entire genomes of both parents are fully represented between the two children.

Single-point crossover has the benefit of preserving many contiguous substrings, or schema, into subsequent generations [92]. But its disadvantages are also readily apparent, such as the tendency of this method to break longer schemata, which are disrupted at the crossover point. If the crossover point occurs at a locus in the middle of a highly desirable

Algorithm 2.12: An algorithm implementing single-point crossover at a randomly selected point, or locus, along the chromosome.

```
Data: A pair of parent individuals $P_1$ and $P_2$ with chromosomes of length $n$
Result: A pair of individuals $C_1$ and $C_2$ with chromosomes of length $n$ representing
        the new children
begin
    $C_1, C_2 \leftarrow \emptyset$
    $crossoverPoint \leftarrow$ randomInt($0..n-2$)
    foreach $locus \in (0..n-1)$ do
        if $locus \leq crossoverPoint$ then
            $C_1[locus] \leftarrow P_1[locus]$
            $C_2[locus] \leftarrow P_2[locus]$
        else
            $C_1[locus] \leftarrow P_2[locus]$
            $C_2[locus] \leftarrow P_1[locus]$
        end
    end
    return $C$
end
```

schema in the first parent, then that schema will likely be corrupted when genes are copied to the child from the second parent, overwriting that portion of the schema that occurs after the crossover point. This "positional bias" therefore favors short, low-order schemata as building blocks over longer ones [5][61]. The tendency for bit positions to affect the preservation of desirable schemata, and the desire to free specific genes from a fixed location or to allow functionally related bits to be more closely located, was one of the reasons for the proposal of an inversion operator by Holland [90].

Similarly, single-point crossover suffers from an endpoint bias in that the crossover point always allows the one endpoint of each parent to remain unaltered: specifically, the first endpoint gene for the first parent and the last endpoint gene for the second parent [126, 128-129].

Multi-point Crossover. To combat some of these undesirable effects of crossover, two-part and multi-part crossover came into use. In two-part crossover, an additional crossover point is selected and the genes from the second parent that occur between the

Algorithm 2.13: An algorithm implementing two-point crossover at randomly selected points, or loci, along the chromosome.

```
Data: A pair of parent individuals $P_1$ and $P_2$ with chromosomes of length $n$
Result: A pair of individuals $C_1$ and $C_2$ with chromosomes of length $n$ representing
        the new children
begin
    $C_1, C_2 \leftarrow \emptyset$
    $L_1 \leftarrow \text{randomInt}(0..n-3)$
    $L_2 \leftarrow \text{randomInt}(L_1..n-2)$
    foreach $locus \in (0..n-1)$ do
        if $locus \leq L_1$ then
            $C_1[locus] \leftarrow P_1[locus]$
            $C_2[locus] \leftarrow P_2[locus]$
        else if $locus \leq L_2$ then
            $C_1[locus] \leftarrow P_2[locus]$
            $C_2[locus] \leftarrow P_1[locus]$
        else
            $C_1[locus] \leftarrow P_1[locus]$
            $C_2[locus] \leftarrow P_2[locus]$
        end
    end
    return $C$
end
```

crossover points are copied to the child *(see Algorithm 2.13)*. Or, as Beasley et al. describe, one can envision the chromosome as a circle of genes with the crossover points defining an arc along which the genes of the second parent are copied [19]. Mitchell suggests that two-point crossover is less likely to disrupt longer schemata [126, 129].

Multi-point crossover is conceptually similar, with merely a larger number of crossover points selected. The number of crossover points can vary, with a common approach being to increase the number of points as the length of the genome increases.

Uniform Crossover. In uniform crossover, sometime referred to as random crossover, a coin toss determines from which parent each gene will be copied. A probability factor determines the likelihood that a gene will come from one parent or another. When the probability factor is 0.5, there is an equal chance of any given gene being contributed by

Algorithm 2.14: An algorithm implementing pure uniform crossover, giving equal probability of any given gene coming from either parent.

```
Data: A pair of parent individuals $P_1$ and $P_2$ with chromosomes of length $n$
Result: A pair of individuals $C_1$ and $C_2$ with chromosomes of length $n$ representing
        the new children
begin
    $C_1, C_2 \leftarrow \emptyset$
    foreach $locus \in (0..n-1)$ do
        $cointoss \leftarrow$ randomBoolean()
        if $cointoss$ then
            $C_1[locus] \leftarrow P_1[locus]$
            $C_2[locus] \leftarrow P_2[locus]$
        else
            $C_1[locus] \leftarrow P_2[locus]$
            $C_2[locus] \leftarrow P_1[locus]$
        end
    end
    return $C_1, C_2$
end
```

either parent and this is referred to as pure uniform crossover *(see Algorithm 2.14)*. Because of the equal probability that no two adjacent genes in any parent will be chosen to be copied to the child, pure uniform crossover tends to destroy valuable schemata in genomes with some structure or relationship between adjacent or nearby genes [164].

There is some argument, however, that uniform crossover, which by its nature tends to result in a more diverse exploration of the search space, is a better alternative to single- and multi-point crossover approaches which tend to better preserve longer schemata [41].

Half-Uniform Crossover. Half-uniform crossover (HUX) is a variation, usually applied to binary genomes, in which exactly half of the non-matching genes between the parents are selected for crossover. This is accomplished by computing the Hamming distance [83], which is the number of differing bits between the two binary genomes, and then dividing that by two. This becomes the number of differing bits that are selected for crossover, and then a coin toss is usually used to determine from which parent each selected gene is taken [21].

Algorithm 2.15: An algorithm implementing parameterized uniform crossover with a probability factor, p_0, as an argument biasing the probability of any given gene being chosen from one parent instead of the other.

```
Data: A pair of parent individuals $P_1$ and $P_2$ with chromosomes of length $n$ and a
      probability $p_0$ of any gene coming from parent $P_1$
Result: A pair of individuals $C_1$ and $C_2$ with chromosomes of length $n$ representing
        the new children
begin
    $C_1, C_2 \leftarrow \emptyset$
    foreach $locus \in (0..n-1)$ do
        $rand \leftarrow$ randomDouble(0..1)
        if $rand \leq p_0$ then
            $C_1[locus] \leftarrow P_1[locus]$
            $C_2[locus] \leftarrow P_2[locus]$
        else
            $C_1[locus] \leftarrow P_2[locus]$
            $C_2[locus] \leftarrow P_1[locus]$
        end
    end
    return $C_1, C_2$
end
```

Parameterized Uniform Crossover. Parameterized uniform crossover, as proposed by Spears and De Jong, extends the concept of uniform crossover by introducing a probability parameter, P_0, that denotes the probability of swapping the alleles of the two parents [164]. A typical probability of swapping is in the range $0.5 \leq p_0 \leq 0.8$ [126]. The probability argument allows control over the rate of swapping and, therefore, allows selection of rates that would tend to be less disruptive than pure uniform crossover *(see Algorithm 2.15)* [164].

Three-Parent Crossover. Three-parent crossover is usually applied to binary genomes and requires the selection of three individuals, usually randomly, as parents. During crossover, for any given locus the values of the first two parents' gene are compared and, if they are identical, then that gene is copied to the child. If the first two parent's genes differ, then the third individual's gene for that locus is used. This results in a child individual where each gene has the value that occurs most often among the three parents *(see Algorithm 2.16).*

Algorithm 2.16: An algorithm implementing three-parent crossover, in which any given gene in the child has the value that occurs most often among the same genes in the three parents.

```
Data: A triplet of parent individuals $P_1$, $P_2$, and $P_3$ with binary chromosomes of
      length $n$
Result: An individual $C$ with chromosomes of length $n$ representing the new child
begin
    $C \leftarrow \emptyset$
    foreach $locus \in (0..n-1)$ do
        if $P_1[locus] = P_2[locus]$ then
            $C[locus] \leftarrow P_1[locus]$
        else
            $C[locus] \leftarrow P_3[locus]$
        end
    end
    return $C$
end
```

Probabilistic Crossover. Probabilistic Crossover, not to be confused with Parameterized Uniform Crossover above, is another approach that establishes a threshold to be exceeded by a randomized value below which no crossover at all would occur, providing a method for parents to essentially clone themselves and preserve their genome in the next generation. Whitely describes the same technique, which he terms "probabilistic crossover" to allow some genomes to remain intact from generation to generation [178]. In this approach a mating pool is filled with suitable individuals and one or more are selected at random for reproduction. In order to reduce the possibility that an especially highly fit parent's genotype might be eliminated through recombination and crossover, a stochastic value function and probability threshold, P_c, is used that, when exceeded, will allow crossover to occur. If the threshold is not met, each parent then produces a child identical to itself, with no crossover occurring, thus preserving the parents' genotypes in the subsequent generation. Mitchell suggests that an appropriate probability threshold for crossover is $P_c < 0.7$ [126, 9]. Probabilistic crossover may not always be desirable, and it can sometimes lead to faster convergence [178].

Although parents' genes have a high likelihood of persisting from one generation to the next, over time this becomes less and less probable. As a result, it is possible that the most highly fit individual encountered during the course of the algorithm's run may not survive until the end .

By this repeated process of selection and crossover, it is hoped that, over time, the overall fitness of the population will increase and after many generations a very fit solution can be returned.

2.6.4 Mutation

The final component of Neo-Darwinism is mutation, which accounts for the introduction of new genetic material into a population, or the reintroduction of old genetic material that may have been lost over time through selection and recombination. Mutation is the most common method for introducing new information and for preventing loss of diversity at a given gene [92][126, 23][178, 5].

During crossover and other recombination operations, all the genetic information that an offspring can possess must come from one or more of the parents. Because of this, the amount of genetic diversity in a population can never be greater than what existed in the initial population unless some other process exists that allows new traits and characteristics to enter.

In fact, over time, without a process for creating new information, the population generally becomes less and less diverse due to the convergence effects of most selection methods. Goldberg writes that "...reproduction alone does nothing to promote exploration of new regions of the search space, since no new points are searched" [81, 31]. He observes that selection and crossover "may become overzealous and lose some potentially useful genetic material" and that mutation "protects against unrecoverable loss" of important genes or gene sequences [81, 14].

In a genetic algorithm, mutation can be incorporated by allowing the possibility that during reproduction individual genes inherited by an offspring can be altered, or replaced with another gene. For example, when a gene is inherited, the value of a stochastic function can be compared against a probability of mutation, called the mutation factor. If a probability threshold is exceeded, that gene can be replaced with another allele, chosen from the set of available alleles [153]. Hansen writes that mutation is an "unbiased" method for injecting new information into the population [85, 873].

Mutation should be applied only occasionally, as high mutation rates can in many cases be detrimental and little better than random search. Holland defines a probability factor P_m that determines the frequency with which mutation is applied to a specific gene and suggests this value should be relatively low [91]. Indeed, Holland writes that mutation "generally has a background role" of "supplying new alleles or new instances of lost alleles" [91, 97].

Mitchell suggests that an appropriate probability threshold, P_m, for mutation to occur is $P_m < 0.001$ [126, 9]. This threshold is supported by Goldberg, who suggested a good mutation rate "on the order of one mutation per thousand bit (position) transfers" [81, 14]. However, higher mutation rates, such as $P_m < 0.0333$, have also been suggested by Goldberg [81, 71].

The value of P_m can be misleading, as the exact implementation of the mutation operation plays a factor in the true mutation rate. For instance, with a mutation rate $P_m = 0.001$, we can expect that, on average, one in every 1,000 genes is mutated. If the mutation operation stochastically assigns one of any valid alleles of the gene, then the actual rate of mutation will vary. If binary gene is chosen for mutation, then the mutation operation has an equal chance of assigning a one or zero to that gene. In half the cases, the mutation operation will randomly assign the gene the same value it had previously. The effective mutation rate is only half of value of P_m. If the gene is non-binary and has k alleles, then the mutation operation has a $\frac{1}{k}$ chance of assigning the same gene, resulting in an effective mutation rate of $\frac{k-1}{k}P_m$.

This can be overcome for binary genes by eliminating the coin-toss when selecting the new value and merely applying an inversion or binary NOT operation. For genes with multiple alleles, the mutation operation can be designed to never assign a gene the same value it had previously. In both of these cases, the effective mutation rate then becomes equal to P_m.

2.6.5 Inversion

Inversion is a primitive genetic operation defined by Holland that allows the reshuffling of the order of alleles in a chromosome [92]. Understanding the inversion operation requires comprehending Holland's concept of Schemata, which encode representations of portions of a chromosome. A fundamental point in Holland's treatment of Schemata is that longer Schemata are more likely to be broken or disrupted during crossover or other genetic operations. [92]

The presumed value of inversion is that the position of individual bits, or genes, associated with a positive outcome can be shifted in the chromosome so that bits that are most valuable in producing a highly fit solution are located more closely together. This results in shorter positive schemata that are, presumably, less likely to be disrupted through crossover and therefore more likely to persist from generation to generation.

Inversion is usually implemented by selecting a segment of the chromosome, typically of both random location and length, and then inverting the ordering of the genes in that segment. This may require some modification of the chromosome such that each bit, or gene, is also associated with a value indicating its unaltered position in the chromosome. Performing inversion on a typical bit array, without preserving the position information in the original bits, "is nothing more than large scale mutation" [179]. Holland defines a probability factor P_i that determines the frequency with which inversion is applied to a specific individual. [91, 121]

To accomplish the preservation of gene position, each gene or bit is often encoded as a tuple (a, b) such that the first parameter represents the bit's original position and the second its value. Consider the following equivalent representations of a 10-bit genome:

```
1 0 1 1 0 1 0 0 1 0

(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0) (9,1) (10,0)

(7,0) (3,1) (9,1) (2,0) (6,1) (10,0) (1,1) (5,0) (8,0) (4,1)
```

This encoding permits bits to be reshuffled in any order without losing the original position, and therefore the meaning, of each bit. It is trivial with this method to decode the tuples to recover a properly ordered bit string.

2.6.6 Loss of Diversity and Convergence

Each generation in a genetic algorithm operates by selecting better individuals, at the expense of poorer individuals, and mating those better individuals to produce the next generation. As a consequence, individuals have differing reproduction rates, and a reasonable selection method should result in good individuals having a reproduction rate $\bar{R}(f) > 1$ and poor individuals having a reproduction rate $\bar{R}(f) < 1$ [24].

Therefore, in each succeeding generation, a certain amount of genetic material from poorer individuals is lost. Blickle and Thiele term this phenomenon "loss of diversity," and quantify it as the proportion of individuals not selected during the selection phase, which he terms p_d [23]. Baker describes a similar calculation, which he calls the reproduction rate, or the percentage of individuals that are selected, as $RR = 100(1 - p_d)$. The loss of diversity should be kept as low as possible, Blickle and Thiele write, because a higher loss of diversity increases the risk of premature convergence [24].

Convergence refers to the phenomenon of a population in an Evolutionary Algorithm approaching a homogenous solution that is, usually, not the global optimum [40, 135]. It is generally characterized by most or all of the population being composed of identical or nearly identical solutions such that there is little genetic diversity remaining in the population.

Convergence occurs quickly in EAs that use selection methods with an inherently high loss of diversity, such as tournament selection, truncation selection, and others.

Chapter 3
Literature Review: Metahuristic Search in Geographic Districting Problems

"*There is a popular cliche ... which says that you cannot get out of computers any more than you put in. Other versions are that computers only do exactly what you tell them to, and that therefore computers are never creative. The cliche is true only in the crashingly trivial sense, the same sense in which Shakespeare never wrote anything except what his first schoolteacher taught him to write — words.*"

— Richard Dawkins, *The Blind Watchmaker* [56]

The challenges of geographical optimization problems are many, and include the physical, demographic, social, and political. Typical geographical optimization problems encompass the obvious domains — transportation, defining sales or delivery regions, drawing of political constituencies, school board boundaries, or similar governmental applications — and the less apparent, such as natural resource management, healthcare system boundaries, epidemiological analysis, emergency services areas, logistics and supply, and others [184].

Two general types of geographic optimization problems are those that require the partitioning of an area or region and those that require selection of a subset of geographic units. For each of these, there are two subtypes, depending on whether the problem does or does not have spatial constraints [184]. “Spatial constrains” means the imposition of an additional geographic requirement on a solution, such as the necessity for selected entities to be physically connected (i.e. contiguous), or that they not be adjacent. Other geographic constraints may include the requirement for a specific subunit to be included in a particular district, or the respect or ignoring of natural boundaries.

Geographic districting problems — which are also known as zone design, territory design, or commercial territory design problems — have been described as an extension of the well-known knapsack problem or traditional clustering problems [12][44, 1072]. In particular, the general constraints are observed to be similar to that of a clustering problem with a set of small geographic units $X = x_1, x_2, ..., x_n$ and number of desired zones k in which to apportion each unit x_i. Z_i is the set of units that belong to zone i, then: [12]

$$Z_i \neq \emptyset, \text{ for } i = 1, ..., k, \tag{3.1}$$

$$Z_i \cap Z_j = \emptyset, \text{ for } i \neq j, \tag{a}$$

$$\cup_{i=1}^{k} Z_i = X \tag{b}$$

In districting problems, the aim is to partition a geographic region into districts, or zones, that satisfy one or more additional constraints that define a desirable zone or set of zones. In political and governmental districting problems, a number of constraints are readily apparent, many of which center on the concept of equity. However, equity can be quantified in a number of ways, many of which can be mutually exclusive: population equality, socioeconomic homogeneity, racial and ethnic homo- or heterogeneity, ethnic and cultural integrity, and others [29].

Regardless of the criteria used to define a desirable zone, districting problems can be described as a multiple-criteria optimized search. There are a large number of potential solutions to this kind of searching problem, including by Tabu Search (TS), simulated annealing, and using Genetic Algorithms.

Bação et al. summarize solutions to districting problems as falling into one of three general categories. The first is those solutions that build zones individually and then attempt to combine them into an aggregate solution. The second includes approaches that take an existing districting plan and modify it by moving small units between districts. The final type includes solutions that generate an entire districting plan at once by allocating all geographic units to a specific zone [12].

Geographic optimization problems are computationally expensive, and many problems in this area fall into the category of NP-complete or NP-hard problems, for which there is no known way to assure the achievement of an optimal solution in reasonable time [12][50]. Therefore, heuristic search techniques would seem to be the best means for obtaining acceptable solutions within a feasible computational time limit [7]. But Altman cautions that heuristic techniques "provide no guarantee of convergence to the optimal district plan in a finite amount of time. At best, they merely good guesses" [7, 91-92].

3.1 Optimized Search for Location and Districting Problems

Early work in location problems, particularly central facilities location, would prove influential in later extrapolations of location problems to districting problems.

Typical location problems are variations on the well-known Weber problem and involve locating one or more warehouses, plants, or other facilities at ideal locations such that the facilities service the surrounding population as efficiently as possible [175]. Specifically, the central facilities location problem consists of determining a subset m of a set of n communities to serve as central locations from which all communities are served in such a way that total travel distances or times are minimized. Typically this is measured as the average of all travel times or distances for the population such that $\bar{s} = \frac{\sum_{i=1}^{n} a_i s_i}{\sum_{i=1}^{n} a_i}$, where a_i is the population of community i and s_i is the cost (i.e. miles or minutes) that individuals in community i must travel to the nearest center [146].

Leon Cooper and Kulin and Kuenne each explored minimum cost solutions to such location problems using a type of gradient descent algorithm, or successive approximations, to converge at an optimum or near-optimum solution [49][112]. Revelle and Swain adopt a similar approach, representing the problem as a matrix of costs between communities and centers that can be solved as a linear programming problem with repeated iterations with the optimal solution found in a small fraction of the time compared to an exhaustive search [146].

In 1965, Hess et al. proposed a method for computer algorithms to provide "a rapid and nonpartisan method...to develop districting plans to meet pre-specified criteria" [86, 998]. They note that the political districting problem is analogous to the "warehouse location" problem described by Baumol and Wolfe in 1958 [18], Keuhn and Hamburger in 1963 [111], and others, but with more complicated constraints. Essentially, a predetermined number of warehouses must be located such that, collectively, they adequately cover all customers in the population with roughly equal apportionment of customers to each warehouse.

The heuristic used in Hess et al.'s approach centered on three criteria: equal population, contiguity, and compactness. Contiguity means, quite simply, that the district be comprised of a single land parcel. Compactness was generally accepted to mean geographically consolidated rather than spread out, although at the time "no geometric measure of compactness (had) been widely accepted" [86]. Population equality was included as a numeric measure equal to the sum of squared distances from each person to the district's center.

Hess et al. employed U.S. Census enumeration districts, an area designed to be covered by a single census taker, as a predefined population unit from which to construct a solution. Among the benefits of using Census enumeration districts are that they already take into

Algorithm 3.1: A Redistricting algorithm proposed by Hess et al. [86, 999-1001]

```
Data: Given $k$ desired districts and $n$ Census enumeration districts (EDs)
Result: Allocation of $n$ Census EDs equally among $k$ districts
1  begin
2      repeat
3          Randomly guess $k$ district centers
4          repeat
5              Assign population equally to centers by minimum-cost transportation
                 algorithm
6              Adjust assignments so each Census ED is contained entirely in a single
                 district
7              Compute district centroids and use as improved district centers
8          until District centers converge;
9      until Until Desired Outcome Reached;
10 end
```

account natural and manmade barriers, such as rivers, highways, railroads, and other features that tend to be desirable for legislative districts [86, 999]. The Hess et al. study focused on the very small state of Delaware, in which districts were comprised of roughly 12,000 people and Census enumeration districts contained typically 1,000 people. But they write that, especially for larger states, census tracts and entire counties could be acceptable base units [86].

Hess et al.'s method was designed to iteratively guess potential district centers, allocate population to the closest center, and compute the resulting centroid for use as a new district center *(see Algorithm 3.1)*. In evaluating the resulting districts, any non-contiguous districts were rejected outright and solutions that exhibited both compactness and population equality, if available, were favored or, if not, the most compact solution within a population deviation limit was selected. Hess et al. observe that while the algorithm does not guarantee convergence, in practice convergence occurred within 10 iterations [86].

Hess et al.'s algorithm can be expressed mathematically, as refined by Bozkaya et al., such that I represents the set of all population units, J is the set of units used as seeds, or district centers, and the cost c_{ij} of assigning unit i to j is a function of the distance between the center of j and the center of i. The number of districts to be created is k. The population of a unit i is p_i and must fall within the interval $[a, b]$. The binary variable x_{ij} is equal to 1 iff (if and only if) unit i is assigned to district center j. This nomenclature allows us to model the problem as follows: [29][86]

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} && (3.2)\\
\text{subject to} \quad & \sum_{j \in J} x_{ij} = 1 \quad (i \in I), && \text{(a)}\\
& \sum_{j \in J} x_{jj} = k, && \text{(b)}\\
& x_{ij} \leqslant x_{jj} \quad (i \in I, j \in J), && \text{(c)}\\
& a \leqslant \sum_{i \in I} p_i x_{ij} \leqslant b \quad (j \in J), && \text{(d)}\\
& x_{ij} = 0 \text{ or } 1 \quad (i \in I, j \in J). && \text{(e)}
\end{aligned}$$

The objective function (3.2) calculates the compactness of a district as the sum of the distances from the district center of each unit that is part of that district. The constraint that each unit can be assigned to one and only one district is shown by (a). The constraint that the total number of districts equal k is shown in (b). Population equality is calculated in (d) as the sum of the populations of all units included in the district, such that the total must fall between the bounds of a and b [29][86].

The general approach and calculations presented by Hess et al. have been reused and modified in many subsequent studies and algorithms addressing districting problems.

A similar formula, presented by Garfinkel and Nemhauser, defines a binary coefficient a_{ij} equal to 1 iff unit i belongs to district j. The cost c_j is assigned to district j and a binary variable x_j is 1 iff district j is selected. This gives a similar expression as follows: [29][75]

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in J} c_j x_j && (3.3) \\ \text{subject to} \quad & \sum_{j \in J} a_{ij} x_j = 1 \quad (i \in I), && \text{(a)} \\ & \sum_{j \in J} x_j = k, && \text{(b)} \\ & x_j = 0 \text{ or } 1 \quad (j \in J). && \text{(c)} \end{aligned}$$

3.2 Common Constraints on Districting Problems

Another approach to heuristic search for political districting utilizes a variation of Tabu Search (TS) that is coupled with a multi-variable optimization function comprised of both hard and soft, weighted constraints.

Commonly used criteria in defining political districts include: geographic continuity; compactness; respect for natural boundaries, such as rivers and bodies of water; population equality; alignment with existing political or administrative boundaries, such as townships and housing subdivisions; socio-economic homogeneity or, conversely, heterogeneity; similarity to existing district boundaries; preservation of community integrity; and equal probability of representation for minority groups [29].

Bozkaya et al. address the common districting constraints described above, but treat some as hard constraints and others as soft ones. In their study, geographic contiguity is a hard constraint while others are soft constraints that are quantified through a weighted, additive multi-criteria function, $F(x) = \sum_r \alpha_r f_r(x)$, where a_r is a weight for a particular criteria and $f_r(x)$ is the value of the objective function for criteria r [29].

3.2.1 Population Equality

Bozkaya et al. define population equality in terms of the population $P_j(x)$ of district $j \in J$ where J is the set of all possible districts. The average district population is $\bar{P} = \sum_{j\in J} \frac{P_j(x)}{k}$. The population of each district must fall within a percentage tolerance $\pm\beta$ of the average population, which is shown by the bound $[(1-\beta)\bar{P}, (1+\beta)\bar{P}]$, where $0 \leq \beta < 1$. This gives a population equality function as follows: [29]

$$f_{pop}(x) = \frac{\sum_{j\in J} max\Big\{P_j(x) - (1+\beta)\bar{P}, (1-\beta)\bar{P} - P_j(x), 0\Big\}}{\bar{P}} \tag{3.4}$$

This equation results in a value of zero if all district populations fall within the tolerance of the average population size, otherwise it returns the sum of the infeasible districts.

Compactness Bozkaya et al. adopt one of two straightforward metrics for measuring district compactness while acknowledging that no metric is perfect. The first is based on total length of all boundary lengths between districts, where $R_j(x)$ is the perimeter of district j and R is the perimeter of the entire territory, which is excluded. The divisor R is used for scaling, and gives the formula: [29]

$$f_{comp1}(x) = \frac{\left(\sum_{j\in J} R_j(x) - R\right)}{2R} \tag{3.5}$$

A second approach compares the perimeter of a district to the circumference of a circle of the same area, where $A_j(x)$ is the area of district j:

$$f_{comp2}(x) = \frac{\sum_{j\in J} \left(1 - \frac{2\pi\sqrt{A_j(x)/\pi}}{R_j(x)}\right)}{k} \tag{3.6}$$

3.2.2 Socio-economic Homogeneity

Socio-economic homogeneity is the degree to which the preferences of individuals in a society tend to be alike, and can be measured by personal income, home ownership, or many other criteria. [76] Bozkaya et al. quantify this as the attempt to minimize the sum, across each district, of the standard deviation $S_j(x)$ of income, which is then divided by the avarage income $\bar{S}$ to provide a dimensionless value: [29]

$$f_{soc}(x) = \sum_{j \in J} \frac{S_j(x)}{\bar{S}} \tag{3.7}$$

3.2.3 Similarity to Existing Plan

Similarity to existing plan can be a desirable metric because incumbent representatives, and their constituents, typically prefer not to have large changes to district boundaries such that a representative no longer represents his or her previously served community. This similarity can be calculated, Bozkaya et al. suggest, using an overlay of the previous boundaries of a district on top of the proposed boundaries for the same district and calculating the areas of those portions that do not overlap: [29, 15]

$$f_{sim}(x) = 1 - \sum_{j \in J} O_j(x)/A \tag{3.8}$$

where A is the entire area and $O_j(x)$ is the largest overlay with a district contained in the new solution x. The authors contend that this approach can be used even if the number of old and new districts differs.

3.2.4 Integrity of Communities

Community integrity is a subjective term, but is generally considered to mean the preservation of a group of individuals who form a “community of interest,” whether on the basis of race, religion, socioeconomic status, or some other criteria. How to preserve communities of interest is also arbitrary, but could be based on "rural/urban divides, shared cultural background, economic interest, ethnic background, demographic similarity, political boundaries, geographic boundaries, and on and on" [95].

To measure integrity of communities, Bozkaya et al. suggest an overlay concept which relies on the minimization of a function calculating the proportion of a community population in respect to the total population of the district: [29, 16]

$$f_{int}(x) = 1 - \frac{\sum\limits_{j \in J} G_j(x)}{\sum\limits_{j \in J} P_j(x)} \tag{3.9}$$

where $G_j(x)$ is the largest population of a given community in district j.

Bozkaya et al. apply each of these metrics as soft, weighted constraints through an objective function $F(x) = \alpha_{pop} f_{pop}(x) + \alpha_{comp} f_{comp}(x) + \alpha_{soc} f_{soc}(x) + \alpha_{sim} f_{sim}(x) + \alpha_{int} f_{int}(x)$ where α_r is a weighting factor and $f_r(x)$ is the value function for criterion r. The weighting multipliers are user-defined, with the exception of α_{pop}, which the authors set initially to 1 and allow to vary during the search to allow some latitude in population equality [29, 16].

Tabu Search (TS) was used as the primary optimization algorithm, in conjunction with an adaptive memory procedure, to iteratively search solutions in the neighborhood of the current solution. Cycling was limited by declaring certain solutions as tabu for a number of iterations, and the search was terminated whenever preset stopping conditions were met.

Bozkaya et al. found that Tabu Search was able to produce solutions that preserve community integrity better than existing plans and maps that are 27% more compact while maintaining populating equality [29].

Another application of Tabu Search to a districting problem involving the monthly assignment of nurses to patients focused on a multi-heuristic approach that factors in workload minimization and continuity of care. Tran et al. partitioned a neighborhood using a Voronoi diagram [10][119] to determine the geographic areas or cells belonging to each patient, giving a set of sites I such that patient $i \in I$ is at the center of his or her own cell. The role of the algorithm then becomes allocating the I patients among a set of K available nurses [169].

Their modified Tabu Search algorithm was compared against both a traditional Tabu Search algorithm and a simulated annealing algorithm on both real-world and randomly generated data. Tran et al. conclude that their application of Tabu Search can, in most cases, allocate nurses to maintain workload equilibrium with smaller workload differences than the comparison algorithms [169].

3.3 Evolutionary Approaches to Partitioning and Districting Problems

Bação et al. write that Genetic Algorithms (GAs) remain largely unexplored in the field of districting problems [12, 342]. However, GAs have been applied extensively in related problems, such as the P-Median problem, cluster analysis, pattern recognition, and other fields. One such approach by Correa et al. explored the use of GAs in solving the P-Median problem, a typical facility location problem that consists of locating p facilities, called medians, which satisfy the needs of n demand points, such that the total cost of serving those demand points is minimized. P-Median problems can be either non-capacitated, such that a facility can service an unlimited number of demand points, or they can be capacitated [50].

The P-Median problem can be represented as an undirected graph, $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is a set of vertices and $\mathbb{E}$ is the set of edges. The problem attempts to find a set of vertices, $V_p \subset \mathbb{V}$, known as the median set, with cardinality p such that the sum of distances between each vertex in the demand set $\{\mathbb{V} - V_p\}$ and its nearest vertex in V_p be minimized: [50]

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j-1}^{n} a_i d_{ij} x_{ij} && (3.10)\\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, ..., n && \text{(a)}\\
& x_{ij} \leq y_j, i, j = 1, 2, ...n && \text{(b)}\\
& \sum_{j=1}^{n} y_j = p && \text{(c)}\\
& x_{ij}, y_j \in \{0, 1\}, i, j = 1, 2, ..., n && \text{(d)}
\end{aligned}$$

where a_i is the demand of vertex j, d_{ij} is the distance or cost from vertex i to vertex j, x_{ij} returns 1 iff vertex i is assigned to facility j, and y_j returns 1 iff vertex j is designated as one of the p medians.

Correa et al. propose a Genetic Algorithm for solving a P-Median problem with a chromosome of length p that encodes the unique ID number of each of the p vertices selected as medians. The fitness is measured by the objective function described in Equation 3.10. Selection was accomplished by a variation on Roulette Wheel Selection (RWS), and elitism ensures only highly fit individuals enter the population. Crossover is accomplished unusually, with two exchange vectors computed for each pair of parents, such that all medians unique to one parent are placed in the exchange vector of the other parent. Then, a random value determines how many of the available medians in each exchange vector are to be swapped between the two parents, producing two new unique offspring. Traditional mutation is accomplished by replacing the mutated gene with any available vertex not already in the current genome [50].

Correa et al. introduce a variation on mutation they term "heuristic hypermutation" that is applied after generation of the initial population and at a probability of 0.05 for each iteration of the running algorithm. Their hypermutation operation consists of selecting a

Algorithm 3.2: An algorithm for "hypermutation" proposed by Correa et al. as part of a solution for the P-Medians problem using genetic algorithms [50].

```
Data: Given a population $P$
Result: The most fit mutation of each individual $X$ in a subset $P' \in P$
begin
    Randomly select $P'$ as 10% of the population $P$
    foreach individual $X$ in $P'$ do
        Let $H$ be the set of facilities not in $X$
        foreach facility $f_i$ in $H$ do
            $Best \leftarrow X$
            foreach gene $g_j$ in $X$ do
                $y \leftarrow$ new individual such that $\{X - g_j\} \cup \{f_i\}$
                if (fitness(Y) < fitness(Best)) then $Best \leftarrow Y$
            end
            if fitness(Best) < fitness(X) then $X \leftarrow Best$
        end
        Insert the new $X$ in the population, replacing the old $X$.
    end
end
```

portion of the population (e.g., 10%) and attempting to improve the fitness of each individual by iterating through all genes in the chromosome, exhaustively replacing that gene with each of the alleles not already in the genome, and then evaluating the resulting fitness for improvement. Each possible allele not already in the genome is tried in each position in the genome, and the allele and position that most improves the fitness is chosen as the replacement. If no mutation results in improvement, then no replacement occurs. This approach is quite similar to that of the traditional steepest-ascent hill-climbing algorithm. The process continues for each individual selected for hypermutation *(see Algorithm 3.2)* [50].

The results of this algorithm were evaluated both with and without hypermutation and compared against a conventional Tabu Search (TS) algorithm. Results shows that the Genetic Algorithm achieves comparable results when compared with Tabu Search in terms of run time, distance or cost minimization, and number of potential solutions evaluated before termination. The GA performs slightly better with hypermutation enabled [50].

Bação et al. explore in a 2005 study the application of Genetic Algorithms to a districting problem, which they term zone design. Citing desirable characteristics in such problems, they also identify population equality, contiguity, and geographical compactness as primary criteria, with the goal of generating solutions that satisfy all three.

To achieve population equality, they employ a minimization function $min \sum_j |P_j - \mu|$, where P_j represents the population of the jth zone and μ (previously expressed as $\bar{P}$ above) is the average population per zone. Compactness is addressed with the presupposition that zones should be as close in shape as possible to a circle. Therefore, a function seeking the minimum of the sums of the distance from the center of each unit to the zone center is used:

$$min \sum_j \Big(|P_j - \mu| + \sum_{i = \in Z_j} d_{ij} \Big) \tag{3.11}$$

where d_{ij} is the distance from the center of unit i to the center of zone j. An alternative measure of compactness is suggested as the product of all distances and the population difference within each zone, which gives an alternative function:

$$min \sum_j \Big(|P_j - \mu| \times \sum_{i = \in Z_j} d_{ij} \Big) \tag{3.12}$$

Finally, they offer a third measure of compactness termed "circumferential compactness" as the sum of the ratios between the square of the perimeter of a zone and its area $\sum_j \frac{pr_j^2}{a_j}$, where pr_j is the perimeter of zone j and a_j is the area of that zone [12, 342-343].

Contiguity is a binary state that is evaluated by a separate algorithm. Contiguity is treated not as an absolute but as a "quasi-hard" constraint by "strongly penalizing non-contiguous solutions, which in practice excludes them" [12].

Bação et al. employed a Genetic Algorithm to solve their zone design problem with one of two encodings, the first of which centers each zone, or district, at the centroid of one of the geographic units that comprise the zone. The second encoding allows zone centers to be placed at any point in the region. Their approach employed ten parallel populations with 25

Algorithm 3.3: A genetic algorithm devised by Bação et al. for solving the zone design problem by repeatedly evolving strings of zone center points. [12, 344]

```
   Data: Given $k$ desired zones, and $n$ geographic units
   Result: The most-fit string $P_{best}$ comprising center points for $k$ zones
1  begin
2      Generate $p$ sets of $k$ center points
3      repeat
4          foreach unit 1..$n$ do
5              Find closest zone center closest and assign unit $n$ to $Z_{closest}$
6              Evaluate fitness of each string $P$ and its assigned units
7              Apply selection, crossover, and mutation operators to create new
                 population
8          end
9      until Generations with no Improvement $\geq$ 5000;
10 end
```

individuals, or strings, per population. Duplicate strings were not allowed, and a very small chance of migration, 0.001, of a string from one population to another was provided for. The optimization functions in Equation 3.11 and Equation 3.12 were used as the heuristic [12, 343-344].

Their final algorithm employed tournament selection; uniform crossover with a probability, P_c, of 0.95; and mutation with a probability, P_m, of 0.001 *(see Algorithm 3.3).* Elitism was employed to ensure the best individual(s) propagate into subsequent generations, and the stopping criteria was defined as 5,000 generations without further improvement [12, 344]. Results showed that the Genetic Algorithm yielded better results than other heuristic approaches, but results were strongly affected by encoding and optimization functions. Adopting a more flexible encoding of zone centers allowed the algorithm to perform better than co-locating zone centers with geographic unit centers [12, 347].

Graph partitioning problems are closely related to districting problems, in that given a graph of connected nodes, each node can be considered to represent one geographic unit and the partitioning of the graph into subgraphs containing nodes is conceptually similar to partitioning a region into districts composed of individual geographic units.

A graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{1, 2, ..., n\}$ is a set of n vertices or nodes and $\mathbb{E} = \{e_{ij} | i, j = 1, 2, ..., n; i \neq j\}$ is the set of edges with e_{ij} representing a connection between nodes i and j. Edges connecting nodes have a weight, w_{ij}, which typically quantifies some cost or distance for the connection between the nodes. The graph G is "directed" if $w_{ij} \neq w_{ji}, (i \neq j)$, otherwise it is "undirected" [54].

Partitioning a graph $\mathcal{G}$ means grouping its n nodes into k non-empty, disjoint sets such that the set of nodes in each set satisfies one or more constraints or objective functions that define a desirable set. Typically, graph partitioning problems employ multiple criteria simultaneously and are therefore considered a multi-objective optimization problem that is NP-complete for $k \geq 2$ [54].

Datta et al. explore the ability of a modified, Non-Dominated Sorting Genetic Algorithm, a type of multi-objective Evolutionary Algorithm, to address the graph partitioning problem. Their approach, which accommodates multiple objective functions simultaneously, shows preliminary success in three out of four test cases. Three criteria for the objective functions are used, including: minimizing the new loss in edge values when considering two nodes for inclusion in the same zone; ensuring similar zone size by minimizing the difference in number of nodes between zones; and promoting compactness by minimizing the spread of a zone in any single direction [54, 626-627]. Other constraints considered by the authors include: an integrity constraint ensuring each node is assigned to one and only one zone; contiguity or connectedness of all nodes within a zone; that the number of zones falls with a predefined range $k_{min} \leq k \leq k_{max}$; and ensuring that the size of a zone falls within a certain range $n_{k_{min}} \leq n_k \leq n_{k_{max}}$.

Initial zone creation is accomplished by first selecting a single node to include in a zone and then expanding it to include neighbors that are not already part of another zone. Crossover is accomplished by generating a new child chromosome by selecting an arbitrary zone from parent A and inserting that zone into the chromosome of parent B to produce a new offspring, after reallocating any overlapping zones or orphaned nodes as a result of the crossover. Mutation randomly shifts the boundary of a zone by allocating one of its nodes to a neighboring zone, thus preserving the integrity constraint described above [54]. The authors conclude that in limited test cases the algorithm is able to find optimal, if known, or near optimal solutions for partitioning test graphs.

Chou et al. used a combination of an evolutionary algorithm and human assistance to subjectively develop and refine an objective function in order to solve a redistricting problem involving political wards in Philadelphia. Their approach investigated the ability of a novel genetic algorithm to partition ten council districts out of the city's 66 wards, each of which is further subdivided for a total of more than 1,300 geographic subunits. Their algorithm employed an objective function to calculate district compactness and employed mutation only, with no recombination of parents at all, to produce offspring [44].

The objective function gauges compactness as a minimization of intra-district distance between subunits and the center of the district. Contiguity and population size are treated as soft constraints that, if violated, result in a penalty to the fitness function. Population size was allowed to vary within an adjustable "slack" range of 5% without penalty. The penalty for breaking the contiguity constraint was harsh, effectively preventing non-contiguous solutions from entering subsequent generations [44].

Chou et al. ignore recombination completely and employ only mutation to explore the solution space. Their approach, which they term *neighborhood mutation*, involves random mutation of a gene representing a ward's district assignment by replacing it with an allele from the set of wards adjacent to the current district. This approach allows mutation to generate child solutions that are much less likely to violate the contiguity constraint. Two mutation rates are employed, with each member of the current generation being mutated a number of times, typically six, to produce multiple offspring. For half of these offspring, a standard rate of $p_{m_{1-2}} = 0.01$ per locus is used, but for the other half a much higher rate of $p_{m_3} = 0.15$ is set. The resulting pool of parents and children, which is seven times the normal population size, is then evaluated for fitness and culled [44].

The algorithm was run for 100 iterations with 2,000 generations for each iteration, which resulted in the discovery of 116 valid districting solutions. These solutions were then presented to a number of subjects who were presented with pairs of potential districting plans and subjectively rated or voted on the more desirable of the pair. Chou et al. call this approach, which has the goal of using subjective human input to inform on the fitness of potential solutions Interactive Evolutionary Computation (IEC). The goal of IEC is to incorporate human input in order to devise or test a potential fitness function that can, in future trials, be used in place of further human input.

IEC algorithms "characteristically (rely) on the judgements of subjects to assess the fitnesses of the solutions encountered in runs of an evolutionary algorithm," they write, and can be valuable in "affording discovery of designs and solutions by evolutionary processes that otherwise could not be well directed for lack of calculable fitness functions" [44, 1074].

A drawback of IEC is that it is labor intensive, requiring humans to evaluate alternatives, essentially performing manual fitness computations. Takagi suggests that 10 to 20 generations of evaluations is usually the most a person can handle [166]. The fatigue problem associated with using human labor in IEC algorithms may be mitigated, Chou et al. write, by using subjective judgements to develop or test a computational fitness function. This function, once validated, could then be used in place of a conventional fitness function in an evolutionary algorithm. Chou et al. call this a validated surrogate fitness (VSF) function [44].

Chapter 4
Solving Districting Problems with Evolutionary Algorithms and Mitigating the Effects of Contiguity Constraints

“*Despite how it’s portrayed in books and movies, artificial intelligence is not a synthetic brain floating in a case of blue liquid somewhere. It is an algorithm – a mathematical equation that tells a computer what functions to perform...*”

— Jeff Goodell [80]

4.1 Constructing an Evolutionary Algorithm to Solve Districting Problems

To evaluate the ability of Evolutionary Algorithms (EAs) to solve real-world districting problems, we first constructed a sample data set consisting of United States Census Data for portions of the State of Alabama. Specifically, we chose sample data to mimic United States Zone Improvement Program areas (i.e. “ZIP” codes) that are fully or partially contained within Jefferson County, Alabama. This area, which comprises much of the downtown and metropolitan Birmingham area, consists of 59 ZIP code zones and is sufficiently large to provide a suitable test set but also still of manageable size. However, census data does not address ZIP codes in a one-to-one manner, especially for rural areas, so ZIP Code Tabulation Areas (ZCTAs), which closely approximate zip codes, were substituted.

Each element in the data set is comprised of the following data points: ZIP Code number; county; municipality (if any); population; and a Keyhole Markup Language (KML) fragment representing the polygon or polygons that define the physical boundaries of the individual parcel. An extensive list of demographic, socioeconomic, and other data is available to be associated with each of these data elements, but have been omitted for the purposes of the preliminary research.

Defining the borders of neighboring zip code parcels could theoretically be accomplished computationally by exhaustively comparing the edges contained in the KML fragments that define the boundaries of each parcel, but for the purposes of this preliminary research, an adjacency matrix has been manually constructed that enumerates for each ZIP code parcel the set of all adjoining parcels.

4.1.1 Construction of Algorithm and Initial Population

The construction of the initial population is trivial and can be accomplished randomly if the geographic contiguity of each district is not required. However, if contiguity is necessary, then the initial, random assignment of parcels to districts would result in a starting population composed entirely, or almost entirely, of invalid candidate solutions.

Therefore, in order to seed the initial population with a set of valid candidate solutions, an algorithm for creating entirely contiguous districts while still maintaining an element of randomness in the composition of those districts is necessary.

If the algorithm for generating a structured, but still randomized, initial candidate solution can be well-constructed, we do not see a problem with using such an algorithm to seed the generation zero population. Hofstadter pointed out that purely random combination is not analogous to nature, where the "chance is infinitesimal that a random combination ... will survive" [88, 662]. Recall also that Fogel stated that it is indeed "reasonable to

Algorithm 4.1: A generalized pseudocode describing the construction of a geographically contiguous initial candidate solution.

```
1 For each $n$ desired initial solutions
2   Select $m$ unique parcels at random and assign each to a district numbered $1..m$.
3   While there are still unassigned districts
4     For each district $1..m$
5       Select at random an unallocated parcel that adjoins the parcels in the district
6       Add the selected unallocated parcel to the district
7   Add the resulting district map to the population
8 Return the initial population with $n$ members
```

incorporate domain-specific knowledge into an algorithm" when constructing a specialized evolutionary algorithm [63, 5].

To accomplish seeding the initial population with entirely valid candidate solutions for problems that require contiguity, a constructor algorithm was been designed that selects n unique initial parcels to serve as center points around which the n desired districts are to be assembled. Next, in a round-robin fashion, for each of the n districts an unallocated, adjoining parcel is added to the district until there are no remaining unallocated parcels. In the case that no unallocated parcels can be added to a particular district, then that district is skipped. This approach is expressed as generalized pseudocode in Algorithm 4.1 and is depicted algorithmically in Algorithm 4.2.

4.1.2 Algorithm Implementation

The algorithm employed can support a varying population size, with a typical population size of between 50 - 250 candidate solutions. The mating process consists of the operation of selection, followed by crossover, with a random chance of mutation occurring during crossover. The selection operation is performed through $k = 2$ tournament selection, which requires that two different competitors are chosen at random from the population, and the competitor with the higher fitness value is designated as the winner of the competition and therefore becomes a parent. This process is repeated twice, resulting in two parents, and is depicted in Algorithm 4.3.

Algorithm 4.2: An algorithm depicting the construction of an initial population of size n consisting of solutions to a districting problem with a geographic contiguity constraint.

```
Data: A set of geographic parcels $D$
Result: A set $P$ containing $n$ geographically contiguous candidate solutions
begin
    $P(\tau) \leftarrow \emptyset$
    foreach $1..n$ desired initial solutions do
        foreach $1..m$ desired districts do
            $D_m \leftarrow$ a unique parcel as district center
        end
        repeat
            foreach district $D_m$ in $(1..m)$ do
                $P' \leftarrow$ a random unallocated parcel that adjoins $D_m$
                $D_m \leftarrow D_m + P'$
            end
        until no unassigned districts remain;
        $P' \leftarrow \sum_{1..m} D_m$
        $P(\tau) \leftarrow P(\tau) + P'$
    end
    return $P(\tau)$
end
```

In this case, there is a provision to prevent the selection of the same candidate solution as a competitor in both tournaments (which will happen with a frequency of roughly $1/n$, where n is the size of the population). However, it is arguably just as acceptable to implement the algorithm in a manner such that if a parent is highly fit and is selected twice, it can then mate with itself, resulting in an identical clone in the subsequent generation.

Genetic exchange during mating is accomplished through single-point crossover with a randomized crossover point anywhere along the genome such that the resulting child receives the entire portion of one parent's genome prior to and including the crossover point and the portion of the other parent's genome after the crossover point. This process is depicted in Algorithm 4.4.

Mutation is incorporated in the mating process with a randomized likelihood of a genetic mutation occurring once out of every 1,000 gene transcriptions, or $p_m = 0.001$. In the case of

Algorithm 4.3: A simple algorithm for selection of ρ parents employing tournament selection with tournament size k. The algorithm explicitly prevents the selection of a single individual as a parent more than once.

```
Data: A population $P$ of size $\mu$, a desired number of parents $\rho$, and a tournament
      size $k$
Result: A subset $P'$ containing $\rho$ unique parents selected for recombination
begin
    $P' \leftarrow \emptyset$
    foreach $i \in (1..\rho)$ do
        $T \leftarrow \emptyset$
        foreach $j \in (1..k)$ do
            $T' \leftarrow \emptyset$
            while $T$ already contains $T'$ do
                $rand \leftarrow$ randomInt$(0..\mu)$
                $T' \leftarrow P_{rand}$
            end
            $T_j \leftarrow T'$
        end
        $P'_i \leftarrow \max(T)$
    end
    return $P'$
end
```

a mutation, the gene being copied is randomly assigned one of the possible alleles between a_1 and a_{max}. No provision is made to prevent a gene selected for mutation from being randomly assigned the value it originally had, which will happen with frequency $\frac{1}{a_{max}}$. This process is also depicted in Algorithm 4.4.

4.1.3 Results

The algorithm described above was executed repeatedly with a varying combination of parameters to assess the feasibility and performance of the main genetic algorithm and its fitness function. For these tests, only a population demographic was used, and the algorithm was designed to partition the sample area into five districts of as nearly equal population as possible. The fitness of a candidate solution was measured, therefore, as the maximum of

Algorithm 4.4: An algorithm employing single-point crossover and incorporating random mutation with a frequency of p_m for each transcribed gene. In the case of a mutation, the transcribed gene is randomly assigned the value of a valid allele in the range of $1..a_{max}$, where a_{max} represents the maximum value of an allele.

```
Data: A pair of parents $P_1$ and $P_2$ with an integer array of length $n$ representing
      genes with valid alleles in the range $1..a_{max}$
Result: A child $C$ with an integer array of length $n$
begin
    $C \leftarrow \emptyset$
    $crossoverPoint \leftarrow$ randomInt($1..n-2$)
    foreach $locus \in (0..n-1)$ do
        $p \leftarrow$ randomDouble($0..1$)
        if $p < p_m$ then
            $allele \leftarrow$ randomInt($1..a_{max}$)
            $C[locus] \leftarrow allele$
        else if $locus \leq crossoverPoint$ then
            $C[locus] \leftarrow P_1[locus]$
        else
            $C[locus] \leftarrow P_2[locus]$
        end
    end
    return $C$
end
```

the difference between each district's population and the average population for all districts in the solution.

In those trials requiring geographic contiguity, candidate solutions with one or more non-contiguous districts were assigned a specific fitness value to represent an invalid solution. The invalid fitness value is such that no valid candidate solution could ever rank lower than an invalid solution. Because of the requirement for geographic continuity in these cases, it is necessary to ensure that the initial population is composed entirely of valid solutions. To accomplish this, Algorithm 4.2 was used. An example generation zero candidate solution created by this algorithm is depicted in Table 4.1 and geographically by the map shown in Figure 4.1.

The precise configurations under which the algorithm was tested include:

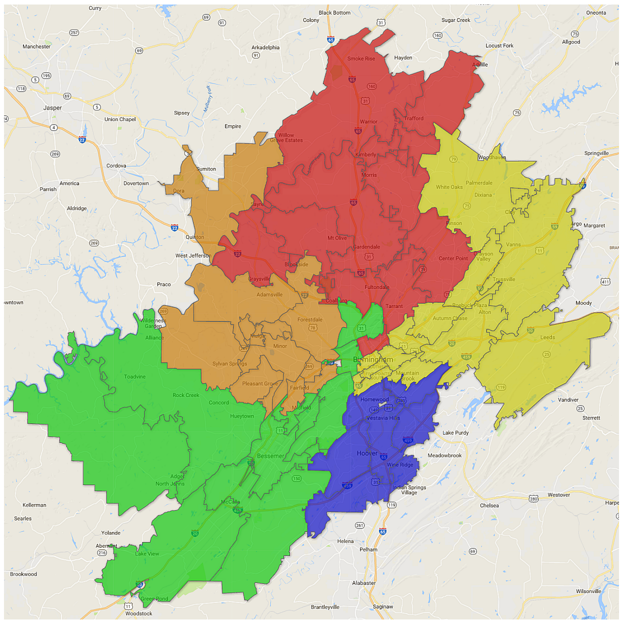
Figure 4.1: A typical map depicting a generation zero, or starting, candidate solution for a simple genetic algorithm to allocate zip code parcels into five contiguous groups, based upon population. Note that each starting group is nearly equal in the number of zip codes each contains, and all zip codes in each group are contiguous.

- Five contiguous districts with $k = 2$ tournament selection, single-point crossover, and random mutation ($p_m = 0.001$).
- Five contiguous districts with $k = 2$ tournament selection, stochastic uniform crossover, and random mutation ($p_m = 0.001$).
- Five non-contiguous districts with $k = 2$ tournament selection, single-point crossover, and random mutation ($p_m = 0.001$).
- with Five non-contiguous districts with $k = 2$ tournament selection, stochastic uniform crossover, and random mutation ($p_m = 0.001$).

Results were consistent over dozens of iterations, with a high similarity among the fitness value of the returned solutions in all cases.

In the case of the two trials requiring contiguous districts, the fitness value of the returned solution usually varied between 650 and 980 individuals. The most-fit solution was

District	**Population**	**Fitness**	**Zip Count**
1	117874	51340	14
2	101432	66782	11
3	169214	66782	12
4	158943	57511	10
5	143852	42240	6

Table 4.1: District populations, fitness value, and number of zip codes in each district for the example generation zero candidate solution depicted in Figure 4.1. Note the wide disparity in population sizes between the zones.

District	**Population**	**Fitness**	**Zip Count**
1	132881	11279	13
2	132797	11363	14
3	144160	11363	12
4	137625	6227	8
5	143852	11055	6

Table 4.2: District populations, fitness value, and number of zip codes in each district for the example 10th generation candidate solution depicted in Figure 4.2. Algorithm variables: $k = 2$ tournament selection, uniform crossover, $p_m = 0.001$.

usually found by the 50th generation, and little, if any, improvement in overall fitness of the population occurred after this. An example 10th generation candidate solution, with a relatively inferior quality solution, is depicted in Table 4.2 and geographically in the map shown in Figure 4.2.

A typical near-optimal solution for contiguous districts, returned by the 50th generation, is depicted in Table 4.3 and geographically in the map shown in Figure 4.2.

District	**Population**	**Fitness**	**Zip Count**
1	137690	1564	15
2	139055	1365	13
3	139254	199	10
4	138438	748	9
5	139221	1531	8

Table 4.3: District populations, fitness value, and number of zip codes in each district for the example near-optimal contiguous candidate solution depicted in Figure 4.3. This solution was repeatedly discovered during multiple runs by the 50th generation. Algorithm variables: population size 10000, $k = 2$ tournament selection, uniform crossover, $p_m = 0.001$.

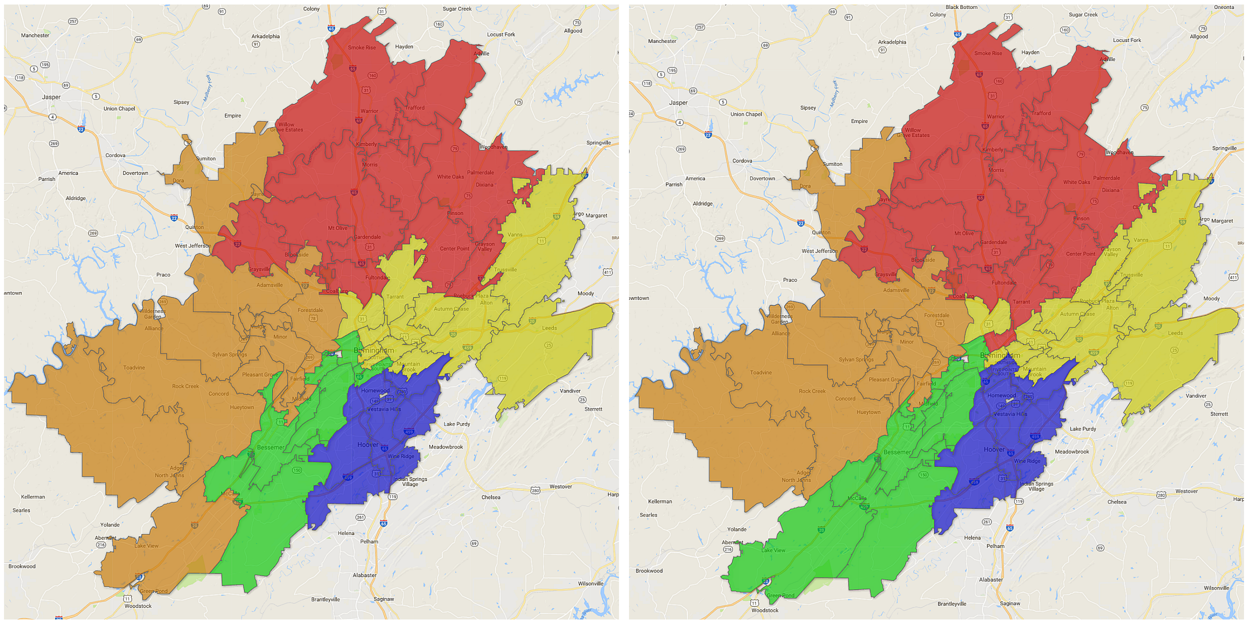
Figure 4.2: A map depicting a much improved candidate solution after only 10 generations, left, and after 50 generations, right, in which zip code parcels are distributed in five districts of contiguous parcels. Each district is created with the goal of having a total population that is as close as possible to the populations of each of the other districts. Note that the number of zip codes in each district varies greatly.

For those trials not requiring geographic contiguity for all zones in each district, the quality of the final solution varied more widely. These trials were typically allowed to run much longer than the trials above, since the requirements for geographic contiguity above quickly eliminated a large number of schemata from the population. In the non-contiguous trials, the quality of candidate solutions often continued to improve even past 100 generations, with some runs still showing improvement near the 300th generation.

For the non-contiguous trials, some returned solutions showed fitness values as high as 250, but a more typical value would fall between 50-95. However, in one *extraordinary* case, a near-optimal solution was returned with a fitness value of only 3.2, representing an average population difference of less than four individuals between all five districts. Any solution with such a negligible difference between its result and the theoretically perfect result of zero population difference between districts would be deemed a success and, although it is impossible to prove whether any single solution provided by a genetic algorithm is optimal,

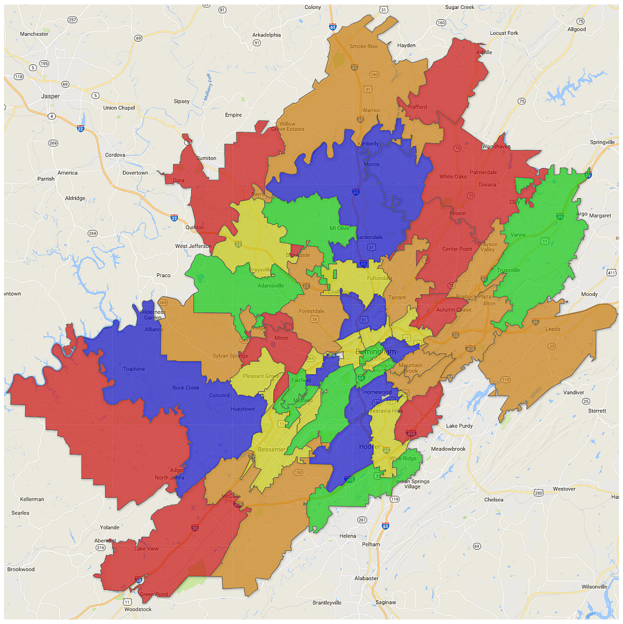

Figure 4.3: A map depicting a near-optimal unconnected candidate solution after 500 generations, in which zip code parcels are apportioned into five districts of non-contiguous parcels. Each district is created with the goal of having a total population that is as close as possible to the populations of each of the other districts. Note that the number of zip codes in each district varies.

it would be foolish to continue searching for a more fit solution once such a solution has been found.

A more typical example solution, with an average population difference in the 70s, is depicted in Table 4.4 and geographically in the map shown in Figure 4.3.

Data from multiple executions with varying population sizes and run (generation) lengths shows that when using $k = 2$ tournament selection there is a very rapid tendency for

District	**Population**	**Fitness**	**Zip Count**
1	138768	88	14
2	138680	88	11
3	138750	70	12
4	138763	83	9
5	138697	71	9

Table 4.4: District populations, fitness value, and number of zip codes in each district for the example 500th generation near-optimal candidate solution depicted in Figure 4.3. Algorithm variables: $k = 2$ tournament selection, uniform crossover, $p_m = 0.001$.

the population to become homogenous, with only random mutation remaining as a factor for introducing new genetic material. Regardless of population size, the population quickly becomes filled with clones of the same solution after only a few score of generations. This results in a quick degradation in the usefulness of subsequent generations. The candidate solution returned is usually good, but the optimization of the returned candidate solution varies widely from run to run. When considering recombination, single-point crossover seems to converge to a better fitness solution faster than stochastic uniform crossover.

Another observation is that increasing the mutation rate above 0.001 produces worse results over time for some trials, most notably those requiring geographic contiguity. Mutation rates as high as $p_m = 0.1$ showed almost no improvement in the population after only a couple dozen generations. Mutation rates of zero allowed the population to rapidly converge on a near-optimal solution, but only if the initial population size was sufficiently large.

The observed results suggest that stochastic mutation, especially in GAs with non-binary alleles, can be a hindrance when used on problems that have an additional structure or relationship between individual genes that would render an otherwise fit solution invalid. Therefore, we conducted additional, controlled testing of a genetic algorithm on a sandbox districting problem to evaluate the effects of contiguity constraints on populations in GAs.

4.2 Controlled Testing of an Evolutionary Algorithm for Solving Districting Problems

To provide a controlled environment in which to refine and evaluate the performance of the genetic algorithm on geographic districting problems, a 25 x 25 grid was constructed with each of the 525 zones randomly seeded with a simulated population in the range of $(200, 25000)$. The sample population data for each zone was generated only once and was kept constant for all subsequent trails, with the simulated data having a minimum population value of 202, a maximum value of 24,985, a mean of 14,317.7, and a total population of 7,516,798 for all zones *(see Table A.1)*.

Other random demographic values were similarly generated for use, if needed, in trails involving a mutli-objective heuristic. The size of the grid was selected to approximate the number of zip codes or ZCTAs in a typical U.S. state.

Two implementations of a 25 x 25 grid were implemented, one which enforced a contiguity requirement for all zones assigned to a district and one which allowed assignment of zones to a district without regard to geographic distance or contiguity. The objective function (i.e. fitness function) of the GA attempts a variation on Wald's "maximin" model [174] to equalize population by minimizing the maximum, or worst, of the differences between the population of a district and the average of the populations of all districts:

$$\text{minimize} \quad \max_{j \in J} \left(\left| P(j) - \bar{P} \right| \right) \tag{4.1}$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad (i \in I), \tag{a}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i \in I, j \in J), \tag{b}$$

$$\text{where} \quad P(j) = \sum_{i \in I} p_i x_{ij} \quad (j \in J), \tag{c}$$

$$\bar{P} = \frac{\sum_{j \in J} \sum_{i \in I} p_i x_{ij}}{k} \tag{d}$$

where $P(j)$ is the sum of the population in district j indicated by (c), $\bar{P}$ is the average of the populations of all districts indicated by (d), x_{ij} in (a) is a constraint function that each zone is assigned to one and only one district, and (b) is a constraint function that is equal to one iff zone i is assigned to district j. This objective function is similar in concept to the algorithm described by Hess et al. in Equation 3.2.

The genetic algorithm was encoded as a 25 x 25 array of integers with values for each index ranging from $0..k$, where 0 indicates the zone represented by the index is unassigned and the values $1..k$ indicating assignment of that zone to the district of the same value. Therefore, each gene in the genome has a set of possible alleles in the range of $1..k$. The value zero is used only during the initialization of the array and is not a valid allele.

4.2.1 Initial Zone Creation

For problems with no requirement for connectedness of districts, initial population construction is implemented by random assignment of the values $1..k$ to each gene, where k is the number of districts in which to partition available geographic units. If fixed district "centers" are required, then the correct district value for zones designated as centers is written after random assignment is accomplished.

Construction of the initial population for problems with a contiguity constraint is accomplished in a manner similar to that proposed by Datta et al. in their work on employing Genetic Algorithms in solving graph partitioning problems [54]. For these cases, an algorithm was constructed that designates k district centers, either arbitrarily or as pre-assigned, and then constructs contiguous districts by iteratively assigning a zone to each district from the set of its unassigned neighbors until there are no remaining unassigned zones. For each district center and its already connected zones, the algorithm determines a set of unassigned zones adjacent to the district and then chooses one arbitrarily to add to the district. This process is completed iteratively for each district, resulting in the addition of one zone to a district during each iteration. If a district has no adjacent, unallocated zones that it can

Algorithm 4.5: An algorithm for randomly constructing a map containing K contiguous districts.

```
Data: A set of $Z$ zones to be divided into $K$ contiguous districts
Result: A set of $K$ districts $D_1, D_2, ..., D_K$ such that all zones in each district are
        contiguous
begin
    foreach $k$ in (1..$K$) do
        Select a zone $Z$ as a district center
        $D_k \leftarrow D_k + Z$
    end
    $k \leftarrow 1$
    while No unallocated zones remain do
        set of neighbors $N \leftarrow \emptyset$
        foreach zone $Z$ in district $D_k$ do
            $N \leftarrow$ set of unallocated zones adjacent to $Z$
        end
        $Z' \leftarrow$ random zone from $N$
        if $Z' \neq \emptyset$ then
            $D_k \leftarrow D_k + Z'$
        end
        if $k \geq K$ then
            $k \leftarrow 1$
        else
            $k \leftarrow k + 1$
        end
    end
end
```

add, then the algorithm continues to the next district. This process continues until there are no remaining unallocated districts *(see Algorithm 4.5)*. The algorithm ensures that the initial population is comprised only of contiguous, feasible candidate solutions that contain roughly the same number of zones, but are not guaranteed to do so.

4.2.2 Effects of Population and Tournament Size Variation

The performance of the Evolutionary Algorithm was measured with a combination of population and selection tournament sizes to assess the effects of population and tournament size on overall fitness of the population as well as the rate of convergence. In a series of trials,

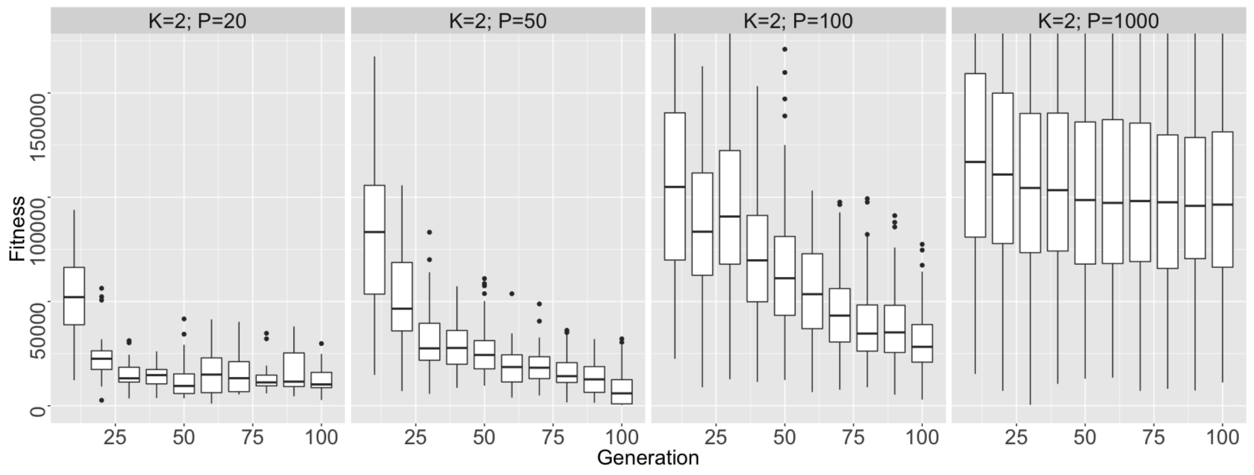

Figure 4.4: Population fitness variation over 100 generations for a 25x25 zone districting problem partitioned into four districts using a population equality heuristic, no geographical contiguity requirement, $k = 2$ tournament selection, single-point crossover, and mutation with probability $p_m = 0.001$. Population sizes, from left, of $\mu = 20$, $\mu = 50$, $\mu = 100$, and $\mu = 1000$ individuals demonstrate that as the population size increases the overall fitness of the population, and therefore the rate of convergence, decreases.

the genetic algorithm was executed with population sizes of $\mu = 20$, $\mu = 50$, $\mu = 100$, and $\mu = 1000$ individuals on a districting problem with no requirement for geographic contiguity. In each trial, a tournament size of $k = 2$, mutation with probability $p_m = 0.001$, and single point crossover was used. Over the course of 100 generations, the fitness of the population was recorded at regular intervals.

A typical trial series is shown in Figure 4.4 and depicts the overall fitness of the population at intervals of every 10 generations. The median value for each measurement is represented by the horizontal line, the upper and lower quartiles are depicted by the boundaries of the boxes above and below the median, and the upper and lower fences are shown by the length of the whiskers. Outliers are depicted individually by dots above or below the fences.

As shown here, increasing the size of the population while keeping the tournament size constant reduces the rate of convergence of the population significantly, leading to less-fit solutions in the same number of generations. With larger population sizes ($\mu > 100$), it may be necessary to increase the tournament size to maintain an acceptable rate of convergence.

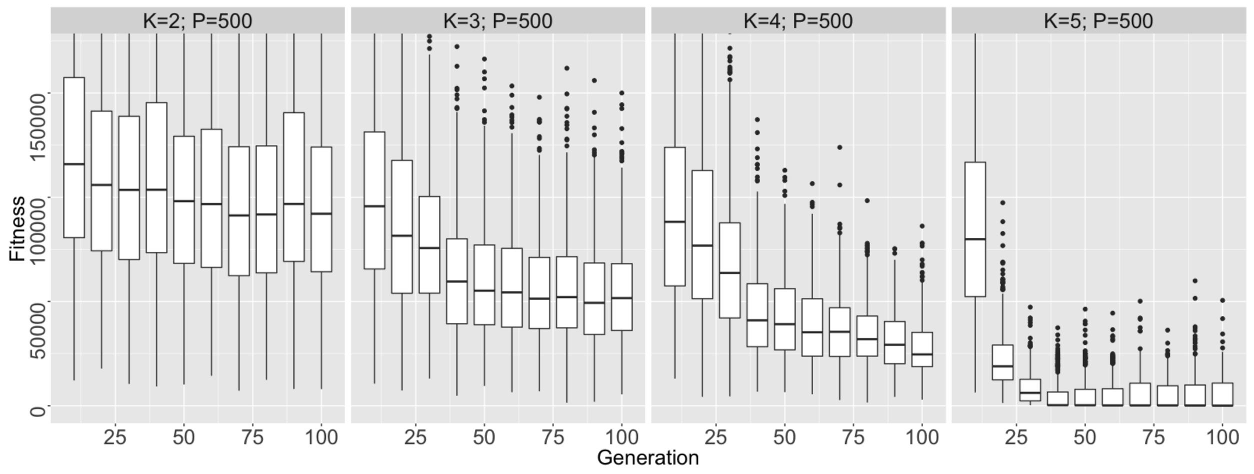

Figure 4.5: Population fitness variation over 100 generations for a 25x25 zone districting problem partitioned into four districts using a population equality heuristic, no geographical contiguity requirement, population size $\mu = 500$, single-point crossover, and mutation with probability $p_m = 0.001$. Tournament size, from left, of $k = 2$, $k = 3$, $k = 4$, and $k = 5$ demonstrate that, for larger populations, an increase in tournament size can improve the overall fitness improvement of the population, and therefore the rate of convergence, from generation to generation.

Figure 4.5 depicts a typical trial series in which the population is kept constant at $\mu = 500$, but the size of the selection tournament is varied from $k = 2$ to $k = 5$. As shown here, the increase of the tournament size allows sufficient chance of selection of highly-fit individuals and elimination of poorly fit individuals; at a minimum, the $k - 1$ worst individuals are guaranteed to not be selected each generation. This allows the population to again show a reasonable rate of convergence. However, as k increases, convergence can occur too rapidly. A reasonable tournament size for large populations ($\mu = 500$ or $\mu = 1000$) would seem to be $k = 3$ or $k = 4$.

4.3 Effects of Hard and Soft Contiguity Constraints on Populations

One of the challenges of districting problems is in determining in which manner to address contiguity constraints, and more specifically, how to handle candidate solutions that violate that constraint. If contiguity is a hard constraint, any candidate solution whose districts are not wholly contiguous is, therefore, infeasible, no matter how well the solution

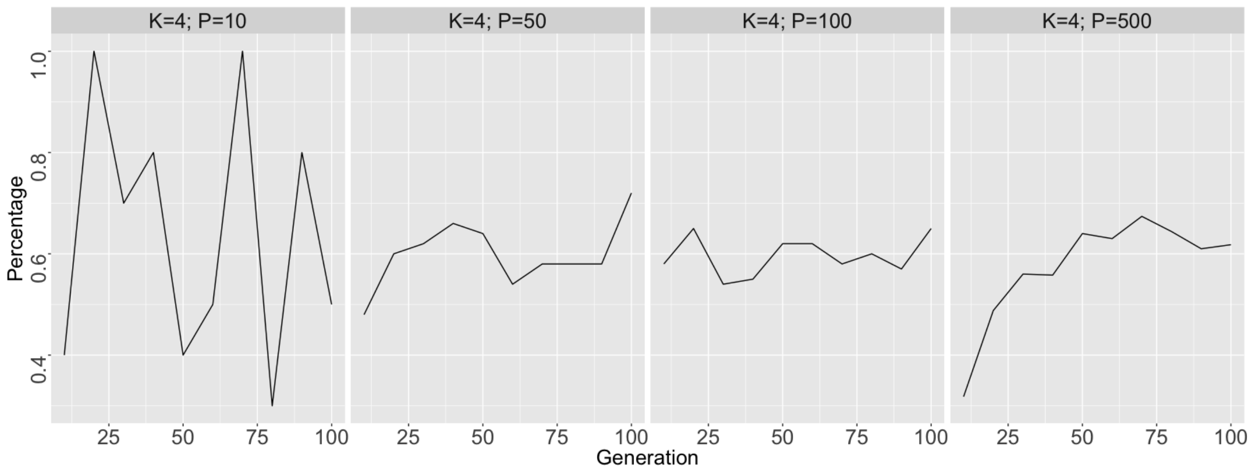

Figure 4.6: Percentage of feasible (i.e. contiguous) candidate solutions in the population over 100 generations for a 25x25 zone districting problem partitioned into four districts with single-point crossover, no mutation, tournament size $k = 4$, population sizes of $\mu = 10$, $\mu = 50$, $\mu = 100$, and $\mu = 500$.

satisfies other, softer constraints. The high incidence of invalid candidate solutions when using an Evolutionary Algorithm to solve a districting problem with a contiguity constraint is demonstrated in Figure 4.6, in which the number of valid, contiguous solutions is shown as a percentage of the overall population for an algorithm employing only mutation to create child solutions.

However, it is wasteful to simply discard a candidate solution that is infeasible because of one or a few zones that violate the contiguity constraint. It is easy to imagine an otherwise highly fit candidate solution that, were a single contiguity error corrected, might already be or could later evolve into the most-fit solution found. Several approaches to addressing contiguity challenges were discussed in Chapter 3, including discarding invalid solution, fitness penalization, and preventative measures.

In fact, there are four generally accepted approaches for dealing with constraints: [47][107]

1. Discarding infeasible solutions entirely, sometimes called the “death penalty.”
2. Employing a penalty function that increases or decreases the fitness value of infeasible solutions.

3. Designing genetic operators to produce only solutions that are feasible.

4. Editing all or part of an infeasible solution's chromosome (i.e. "repair" it).

Here, we implement and compare the performance of three of these methods for addressing candidate solutions with contiguity constraint violations. We consider these methods to fall into three categories: *punitive*, *preventive*, and *remedial.* That is, first we detect and handicap infeasible solutions by assessing a fitness penalty. Next, we design and implement an algorithm for Neighborhood Mutation to reduce the incidence of infeasible solutions. Finally, we design and implement a Local Repair algorithm to detect and correct district contiguity errors.

Each of these three approaches was exercised for 100 trials of an evolutionary algorithm running for 300 generations using a fixed data set. The algorithms employed tournament selection with $k = 2$ to select a single parent and applied no recombination or crossover operation, relying instead exclusively on mutation for generation of new child solutions with a fixed probability of mutation of any given gene of $p_m = 0.001$. This was repeated for each approach using three different population sizes of $\mu = 50$, $\mu = 100$, and $\mu = 500$. In total, 90,000 generations of computations were conducted, involving the creation and heuristic evaluation of 58,500,000 candidate solutions. A sample, highly fit solution is depicted in Figure 4.7.

4.3.1 Fitness Penalization

Fitness penalization is a common tactic in evolutionary algorithms, especially those with multi-objective, weighted heuristic functions [17][47]. In using Genetic Algorithms to solve districting problems with a contiguity constraint, Bação et al. and Chou et al. addressed discontiguous solutions by assessing a fitness penalty that "in practice excludes them" [12][44].

We chose a similar approach and implemented a contiguity detection algorithm that assigns a binary value to each solution determining its adherence to the contiguity constraint

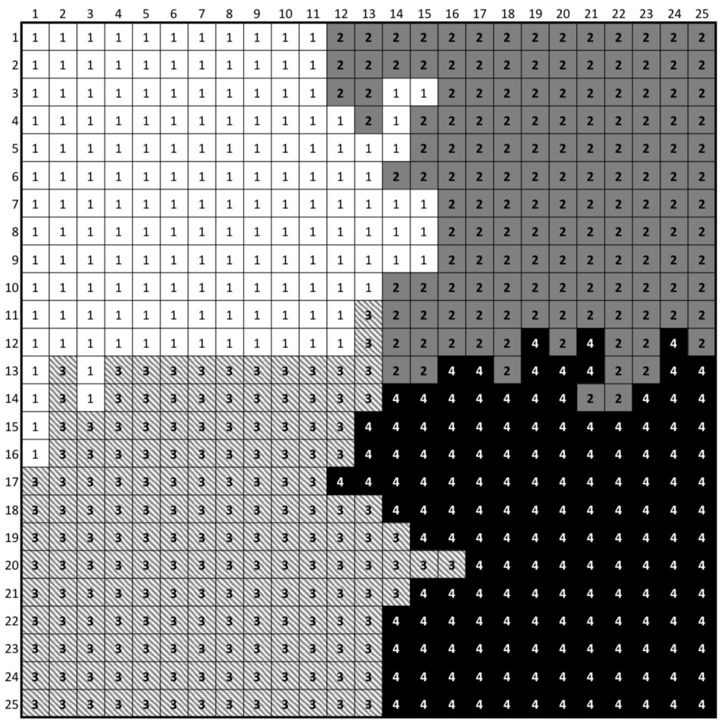

Figure 4.7: A highly fit solution to a districting problem for a problem space of 525 geographic units partitioned into four districts with fixed district centers using a population equality heuristic and a geographic contiguity constraint. The depicted solution was reached after 300 generations with a population size of $\mu = 100$, $k = 4$ tournament selection, single-point crossover, and no mutation. The solution contains a per-district variation from the average population of 125 or less based on a simulated population of more than 7,500,000 among all districts.

(true) or its violation of it (false). In cases that violate the contiguity constraint, the fitness function increases the fitness value of the solution by a fixed value of 50,000, which is significantly higher than any feasible solution in all but the earliest generations of execution.

It is necessary to note that purely random mutation is highly destructive to chromosomes in a districting problem with a contiguity constraint. For subdivision of the 25x25 grid problem into four districts, as done here, a worst case scenario would be a solution in which the area is equally divided into four quadrants, each representing a district. This would result in a solution containing 429 zones that border only zones in the same district. Mutation of the gene for any of these zones would result in an infeasible solution. For the remaining 96 zones, each of which borders at least one zone assigned to another district, any mutation

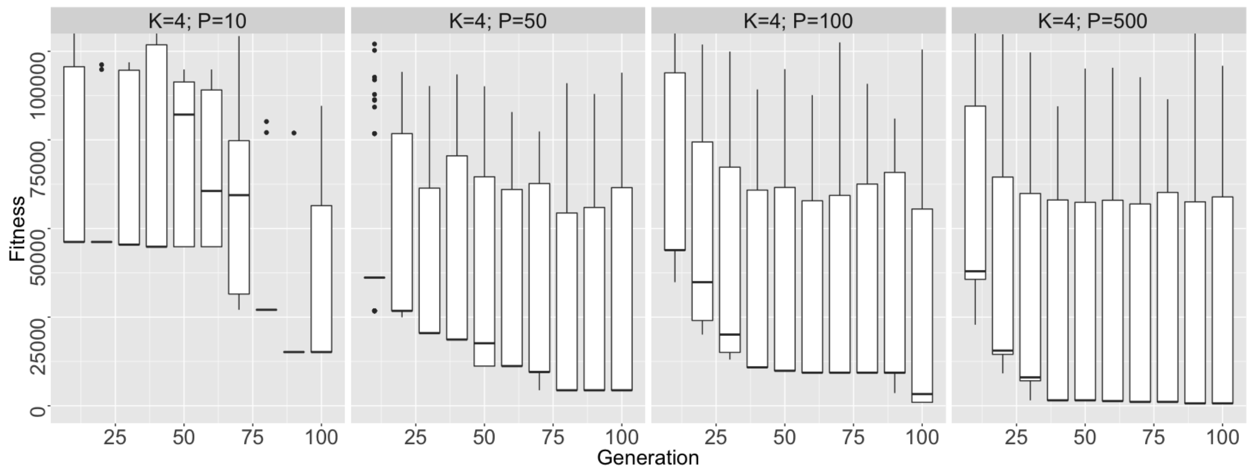

Figure 4.8: Population fitness variation over 100 generations for a 25x25 zone districting problem partitioned into four districts with single-point crossover, no mutation, tournament size $k = 4$, population sizes of $\mu = 10$, $\mu = 50$, $\mu = 100$, and $\mu = 500$, and a population equality heuristic and a geographical contiguity requirement with a fixed fitness penalization for constraint violation of 50,000. The horizontal median bars and small bottom quartile boxes indicate that many highly fit solutions exist in the population, but the large upper quartile boxes reflect the skewing effects of the fixed fitness penalty for a smaller number of infeasible, discontiguous solutions.

would have up to a 50% probability of resulting in an infeasible solution. With a mutation probability of $p_m = 0.001$ and a problem with genome of length 525, there is approximately a 41% chance of each solution undergoing mutation ($p = 1 - \left(1 - 0.001\right)^{525} = 0.4086$). For mutation rates higher than $p_m = 0.001$, which would be expected in a mutation-only evolutionary algorithm, the chance of producing an infeasible solution increases accordingly.

For example, Figure 4.8 shows the effects of a fitness penalty on the distribution of candidate solution fitness by generation for various population sizes. We can see that the upper quartile, indicated by the white rectangles above the horizontal median line, show a wide range of fitness variation due to the fitness penalty. In contrast, the small or barely noticeable lower quartile box, shown below the median line, demonstrates the tight fitness clustering of those candidate solutions that are contiguous and which incur no penalty. It is worth noting that, even with the skewing caused by large numbers of discontiguous solutions, there is still convergence in the algorithm overall, which is demonstrated by the decreasing value of the median line from generation to generation.

The high tendency of mutation to produce infeasible solutions, and thus incur the severe fitness penalty for violating the contiguity constraint, should not be surprising, and indeed has parallels in biology. Considering the enormous number of combinations of alleles in even a simple organism's chromosomes, it is unsurprising that the vast number of mutations possible in nature are detrimental and only the occasional mutation is of benefit to the organism and the species [88]. In pointing out this natural comparison, Holland writes that "in realistic cases, the overwhelming proportion of *possible* variants (all possible allele combinations, not just those observed) are incapable of surviving to produce offspring in the environments encountered" [91, 12]. Hofstadter similarly writes that the "chance is infinitesimal that a random combination of pieces of DNA will code for anything that will survive — something like the chance that a random combination of words from two books will make another book" [88, 662].

Evaluation of algorithm performance with random mutation with probability $p_m = 0.001$ demonstrates that the vast percentage of solutions produced are infeasible *(see Table A.3)*. Over the course of 300 generations, the percent of contiguous solutions in the population falls as low as 17.1% with a population size $\mu = 50$ and has a maximum of 44.51% with population size $\mu = 500$, discounting the first few generations during which the population is still stabilizing. More importantly, the mean fitness values for each generation remain very high compared to the best solution found, demonstrating the skewing of the overall population fitness by the large number of infeasible, penalized solutions. Similarly, the standard deviation among the mean fitness values for each of the 100 trials is high and remains high for population sizes $\mu = 50$ and $\mu = 100$, showing that there is little improvement in the wide range between mean fitness values over the trials. Only with the largest population size, $\mu = 500$, do we begin to see a steady downward trend in the mean fitness value for each generation over time and a similar decrease in the standard deviation between mean fitness values, indicating a greater tendency of agreement in mean fitness values over the 100 trials.

Algorithm 4.6: A Neighborhood Mutation algorithm for preserving contiguity when mutating values in a contiguous districting problem.

```
Data: A set of zones $X$ and a mutation probability $p_m$
Result: A set of zones $X$ with each zone's district assignment having been changed
        to that of a neighboring zone with a probability $p_m$
begin
    foreach district $X_i \in X$ do
        $p \leftarrow randomDouble(0..1)$
        if $p \leq p_m$ then
            Set of alleles $A \leftarrow \emptyset$
            foreach adjacent neighbor $N_j$ of $X_i$ do
                if $N_j.value \notin A$ then
                    $A \leftarrow A + N_j.value$
                end
            end
            $r \leftarrow randomInt(0..Size(A))$
            $X_i \leftarrow A_r$
        end
    end
    return $X$
end
```

However, despite the large percentage of discontiguous solutions, the algorithm still shows convergence for all populations sizes, and returns a best solution, across all 100 trials, with fitness values of 1,444, 653, and 100 for populations sizes $\mu = 50$, $\mu = 100$, and $\mu = 500$, respectively. This would suggest that, despite the presence of a majority of infeasible solutions, there are a sufficient number of valid candidate solutions in the population, especially for the larger population size, to allow the genetic algorithm to sufficiently explore the search space and find near optimal solutions.

4.3.2 Neighborhood Mutation

To reduce the likelihood of mutation resulting in an infeasible solution, Chou et al. proposed the concept of Neighborhood Mutation, in which genes selected for mutation are randomly assigned the value of an allele from an adjacent zone, rather than from all possible alleles [44]. This approach greatly reduces, but doesn't eliminate, the chances of producing

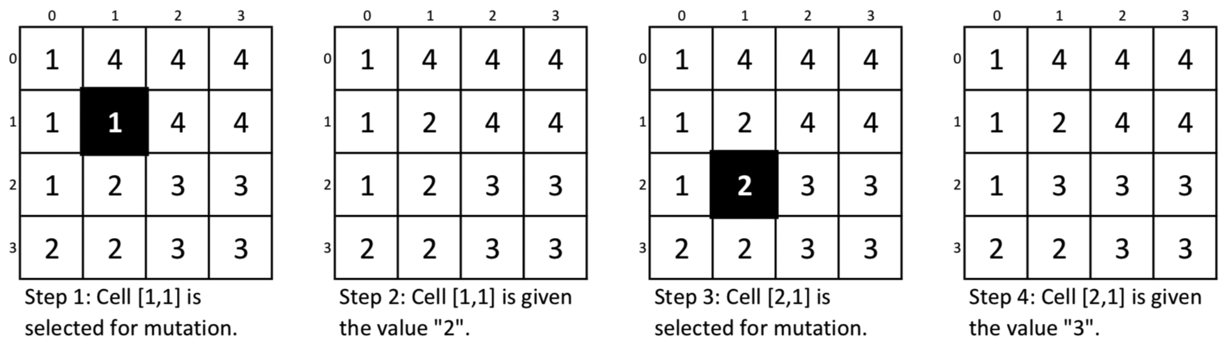
Step 1: Cell [1,1] is selected for mutation.
Step 2: Cell [1,1] is given the value "2".
Step 3: Cell [2,1] is selected for mutation.
Step 4: Cell [2,1] is given the value "3".

Figure 4.9: An illustration of how iterative Neighborhood Mutation can still produce a non-contiguous map. Cell [1,1] is mutated in Step 1 to the value of its bottom neighbor, resulting in the map shown in Step 2. Next, cell [2,1] is mutated in Step 3 to the value of its right neighbor, resulting in the map shown in Step 4 and isolating cell [1,1], causing the discontiguity.

a non-contiguous solution. The Neighborhood Mutation method depicted in Algorithm 4.6 was implemented and evaluated.

Neighborhood Mutation, however, is not perfect. Even with Neighborhood Mutation ensuring that no gene is mutated to an allele that is not among the alleles of adjacent districts, there is still the potential for Neighborhood Mutation to create an infeasible, non-contiguous map. For instance, consider the possibility of a particular cell being mutated to a neighboring district's value such that the new cell forms the tip of a “finger” protruding from the center of its new district. If another cell along the length of that finger is then mutated to a neighboring district value, this can essentially amputate the tip of the finger, forming a discontiguous cell or group of cells that are now isolated from their former district. An example of this process is illustrated in Figure 4.9.

Evaluation of Neighborhood Mutation performance with mutation probability $p_m = 0.001$ demonstrates that the vast percentage of solutions produced — more than 99% in all trials — are contiguous and, therefore, valid *(see Table A.4)*. Over the course of 300 generations, the percent of contiguous solutions in the population was not observed to fall below 99.06%, regardless of population size. This produced both a vastly greater rate of improvement in mean fitness over time compared to random mutation but also the discovery of a higher quality solution over the course of each of the three 100-trial runs. Convergence was observed to occur rapidly, with highly fit solutions being reached in 40-60 generations for the larger population sizes, while the trial with population size $\mu = 50$ did not discover its best solution until almost the 200th generation. The overall quality of the most-fit solutions returned were, in increasing order of population size, 122, 42, and 60.

The standard deviation among the mean fitness values quickly became very tight and grew tighter the larger the population size — barely over 1,000 for $\mu = 50$ and just over 400 for $\mu = 500$. The range of variation among the mean fitness values encountered in each trial was observed to initially vary greatly, but then tighten quickly and within about 50 generations for large populations — or 100 generation for small populations — to reach a stable range. Compared to random mutation, the mean fitness values produced by Neighborhood Mutation over the 100 trials, as shown by these standard deviation values, demonstrates about 1/10th the variation for any given generation as seen in random mutation.

4.3.3 Local Repair

While Neighborhood Mutation is preventative in its approach, we also investigated a remedial method to correct discontiguous solutions once they had occurred. This approach, which we call Local Repair, is applied after the typical genetic operators of selection, recombination/crossover, and mutation have occurred (although, as stated, no recombination or crossover was used in this instance). This approach, very simply, first evaluates the candidate solution for contiguity and assigns the solution a boolean attribute indicating its adherence to the contiguity constraint. If the solution is invalid, the repair mechanism is applied and attempts to evaluate each zone by first determining how many adjacent zones are assigned to the same district. If that number is less than 1, then the zone is deemed to be discontiguous and the repair mechanism will assign it the allele that is most prevalent among its adjacent neighbors. In the event of two alleles being equally prevalent, then the value chosen will

Algorithm 4.7: An algorithm for Local Repair of district contiguity problems by detecting a geographically discontiguous zone and changing its allele to the most prominent allele value in its immediate neighborhood.

```
Data: A set of zones $X$
Result: A set of zones $X$ with any discontiguous zone's district assignment changed
        to the value most common among its neighbors
begin
    foreach district $X_i \in X$ do
        $neighbors \leftarrow \emptyset$
        $connectedNeighbors \leftarrow \emptyset$
        foreach adjacent neighbor $N_j$ of $X_i$ do
            $neighbors \leftarrow neighbors + N_j$
            if $N_j = X_i$ then
                $connectedNeighbors \leftarrow connectedNeighbors + N_j$
            end
        end
        if $Size(connectedNeighbors) < 1$ then
            $N_j \leftarrow$Mode($neighbors$)
        end
    end
    return $X$
end
```

depend on the particular implementation of the Mode function. This method is illustrated in Algorithm 4.7.

Although even more reliable than Neighborhood Mutation in producing contiguous solutions, there remains a very small chance that a solution, such as one with simultaneous, destructive mutations of several adjacent zones, cannot be repaired by this algorithm.

The Local Repair algorithm performs exceptionally well in correcting discontiguous solutions produced by random mutation with probability $p_m = 0.001$, producing valid solutions with very high frequency. In fact, over all trials, the average percentage of contiguous, valid solutions was observed to range as high as 99.96% and never lower than 99.80% *(see Table A.5)*. This represents a reduction in the number of infeasible solutions per generation, on average, to merely 16% of that seen in the already efficient Neighborhood Mutation algorithm.

Local Repair also produced a rate of improvement in mean fitness much greater than random mutation and comparable to that of Neighborhood Mutation. The quality of solutions returned were similarly superior to random mutation and again comparable to Neighborhood Mutation. Convergence was observed to occur rapidly, with highly fit solutions being reached in about 100 generation for all population sizes. The overall quality of the most-fit solutions returned were, in increasing order of population size, 221, 49, and 66.

The standard deviation among the mean fitness values quickly grew tight for all population sizes, but was observed to do so at a slightly slower rate than seen with Neighborhood Mutation, again settling within about 50 generations for large populations and slightly slower for smaller populations. Compared to random mutation, the mean fitness values produced by Local Repair over the 100 trials, as again shown by these standard deviation values, demonstrates slightly less variation than seen in Neighborhood Mutation and, again, about 1/10th the variation for any given generation as seen in stochastic mutation.

4.4 Algorithm Performance on Statewide Analysis

Algorithm performance on real-world, state-wide data was conducted using the state of Alabama and demographic and geographic information from the decennial census [39]. The smallest geographic unit employed is the ZIP Code Tabulation Area (ZCTA), a census unit closely approximating a U.S. Postal Service Zone Improvement Program (ZIP) Code service area [171][172]. There are 642 ZCTAs that are wholly contained within the state of Alabama and account for the vast majority of its land area. Only very small portions of the state are omitted from ZCTA coverage, mostly unpopulated tidal wetlands, national forests or preserves, military reservations, or small tracts that are part of ZCTAs primarily contained within neighboring states. A mapping of Alabama ZCTAs is shown in Figure A.1.

In constructing the framework for the algorithm, geographic data defining the borders of ZCTAs were obtained from the U.S. Census Bureau and extracted as one or more polygons or multi-geometry Keyhole Markup Language (KML) segments for each ZCTA. Although the nature of the geographic polygons would theoretically allow the dynamic evaluation of adjacency between one ZCTA and another, the computational complexity would have significantly increased the execution time of each algorithmic run. Because the ZCTA adjacencies are unchanging and need only be computed once, an adjacency table was manually created that catalogs each zone and its immediate neighbors. The adjacency list is expressed in a similar manner to what would be expected for the representation of a simple, bidirectional graph. The average connectedness of all ZCTAs is 5.52 connections, with a maximum of 16 connections, a median of 6 connections, and a minimum of 1 connection for a handful of small zones completely contained within other zones.

Other geographic units than ZCTAss are available with associated demographic data and could be utilized in lieu of ZCTAs by the algorithm. The most notable among these alternatives are Census Tracts, which are small units encompassing from 1,200 to 8,000 people with an optimum size of 4,000 [170]. The size of a census tract can vary greatly according to the population density of the area being considered. In comparison, Alabama's ZCTAs have an average population of 7444.8 persons, with a maximum population of 50,268, a minimum of 7, and a median of 4028. The choice between census tracts and ZCTAs is considered functionally equivalent. A 1965 study by Hess et al. utilized U.S. Census enumeration districts, a geographic area since replaced by census tracts, that was intended to be covered by a single census worker and contained approximately 1,000 people [173]. The Hess et al. study utilized enumeration districts because it was conducted within the state of Deleware, which is the second smallest state with an area of only 1,981 square miles, and the smaller units were presumably desirable. Hess et al. explicitly write that, for larger states, larger census units and even counties could be utilized in lieu of enumeration districts [86].

The structure of the evolutionary algorithm was strongly influenced by earlier results obtained with a limited set of ZCTA data and on the sandbox experiments using a 25x25 grid with simulated demographic data. The initial experiments on a small set of ZCTA data confirmed the efficacy of a basic algorithm to create and improve successive generations of candidate solutions on both unconnected and connected districting problems. Later controlled experiments using the 25x25 grid sandbox and simulated data explored the effects of variation of population size, tournament selection size, recombination, and mutation on the performance of the algorithm.

Observed results suggest desirable values for population size of $\mu = 100$ and tournament selection size of $k = 3$. The effects of stochastic mutation on the feasibility of candidate solutions — that is, geographic contiguity or discontiguity — show that even typically low levels of mutation ($p_m = 0.001$) can be destructive to a large portion of the population from generation to generation. Although the number of contiguous, and therefore valid, solutions does increase in the population over time, the percentage of valid solutions hovers around 60% with little additional improvement. Therefore, the implementation of Neighborhood Mutation and Local Repair algorithms were employed to prevent and mitigate these effects. Each is able to achieve geographic contiguity in more than 99% of all candidate solutions. Although the Local Repair algorithm is more effective than Neighborhood Mutation, the high reliability of each independently suggests eliminating the need to consider using both together. The effects of various crossover methods are even more destructive. Uniform random crossover is disastrous in almost every instance and single-point and multi-point crossover fare little better, producing such a high percentage of infeasible solutions that further consideration is unwarranted.

In light of these findings, a mutation-only algorithm which employs no crossover, single-parent tournament selection with tournament size $k = 3$, and a population size $\mu = 100$ was used. Neighborhood Mutation was employed in lieu of stochastic mutation, and, in most cases, Local Repair was omitted. The algorithm is designed such that district centers can be either fixed for all iterations or determined randomly at the start of each iteration, but were allowed to randomly float for all iterations and trials. A soft-contiguity constraint, similar to that proposed by Bação et al., Chou et al., and others, was implemented to allow discontiguous solutions into the population, but with a severe fitness penalty.

The algorithm was executed for 100 iterations with a limit of 500 generations in each iteration. In most cases, four separate trials were conducted, each with a varying mutation rate ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$), resulting in the generation and heuristic evaluation of 20 million candidate solutions for each assessment of the algorithm.

4.4.1 Performance on Population Equality Heuristic

First, the algorithm was evaluated on a districting problem using a population equality heuristic for fitness evaluation and a soft contiguity constraint that penalizes discontiguous solutions with the addition of a fixed fitness increase of 50,000.

The algorithm is shown in this case to be effective in obtaining near-optimal solutions on the statewide data set for two-criteria optimized search with a population-based heuristic and a soft contiguity constraint. In each of the four trials, the algorithm produced a reasonable and acceptable solution for districting Alabama's 642 ZCTAs into seven contiguous districts of nearly equal population *(see Figure 4.11)*. Variation in mutation rates between the four trials shows that for all trials the algorithm show a rapid improvement in discovery of highly fit solutions. The observed rate of improvement in both the minimum fitness value *(see Figure 4.10)* and the mean fitness value *(see Figure 4.13)* is most rapid with a mutation rate of $p_m = 0.005$, only slightly faster than with a rate of $p_m = 0.0025$, but performance falters with mutation rates of $p_m = 0.010$ and higher, which alter more positions simultaneously in a child solution's genome.

In each of the four trials, the maximum divergence of any given district's population from the average for all districts is extremely low, with the divergences in the three best trials of only 66, 72, and 78 persons *(see Figure 4.14)*. Only the trial with the highest mutation rate produced a significantly worse divergence from the average per district, finding a candidate solution with a minimum fitness value of 853 persons or less.

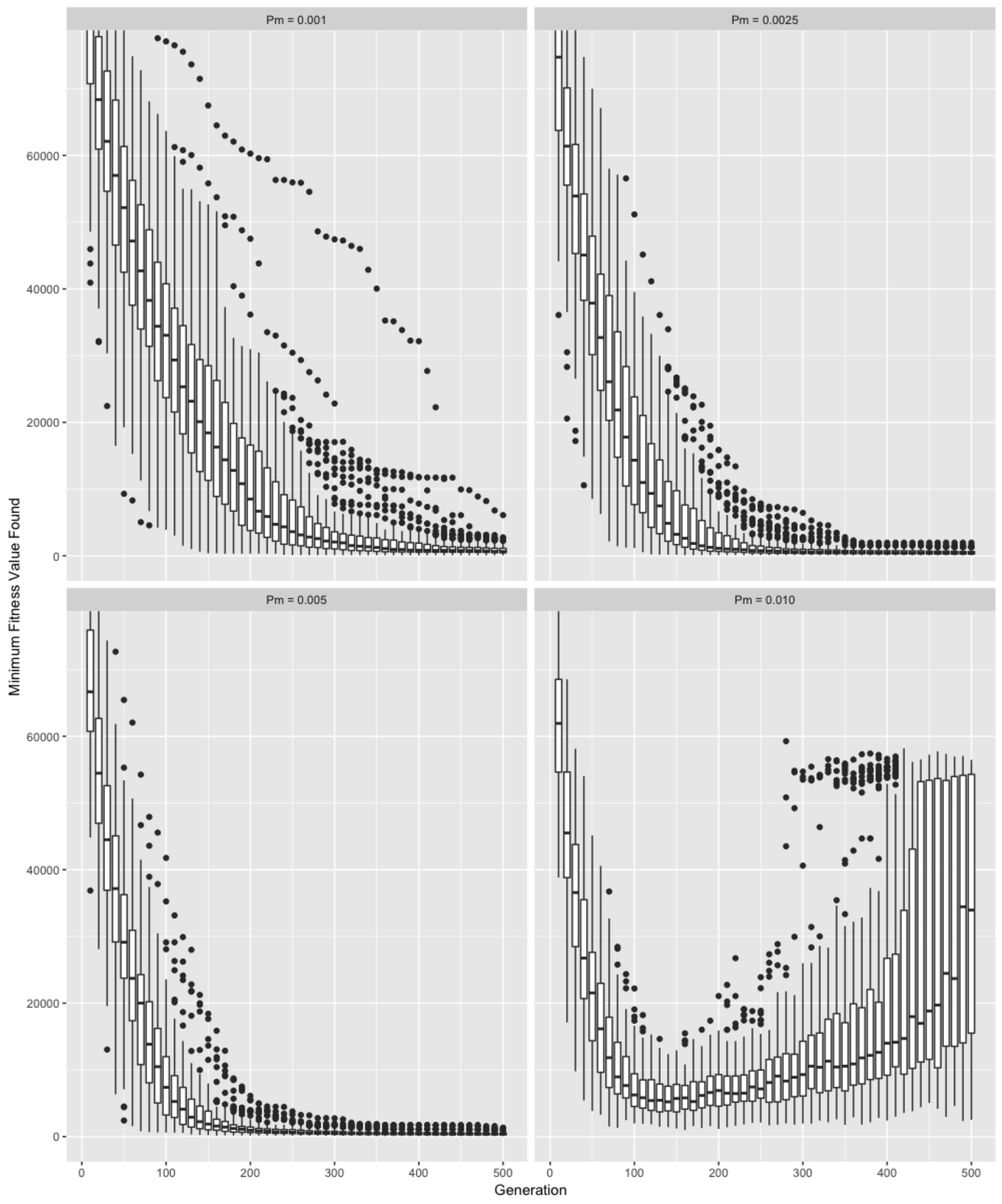

Figure 4.10: Minimum fitness value found during each iteration, by generation, for 100 iterations of evolutionary search for solving a contiguous districting problem dividing the state of Alabama into seven districts. The algorithm employed a population size $\mu = 100$, no recombination or crossover, single-parent tournament selection with tournament size $k = 3$, and Neighborhood Mutation with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$).

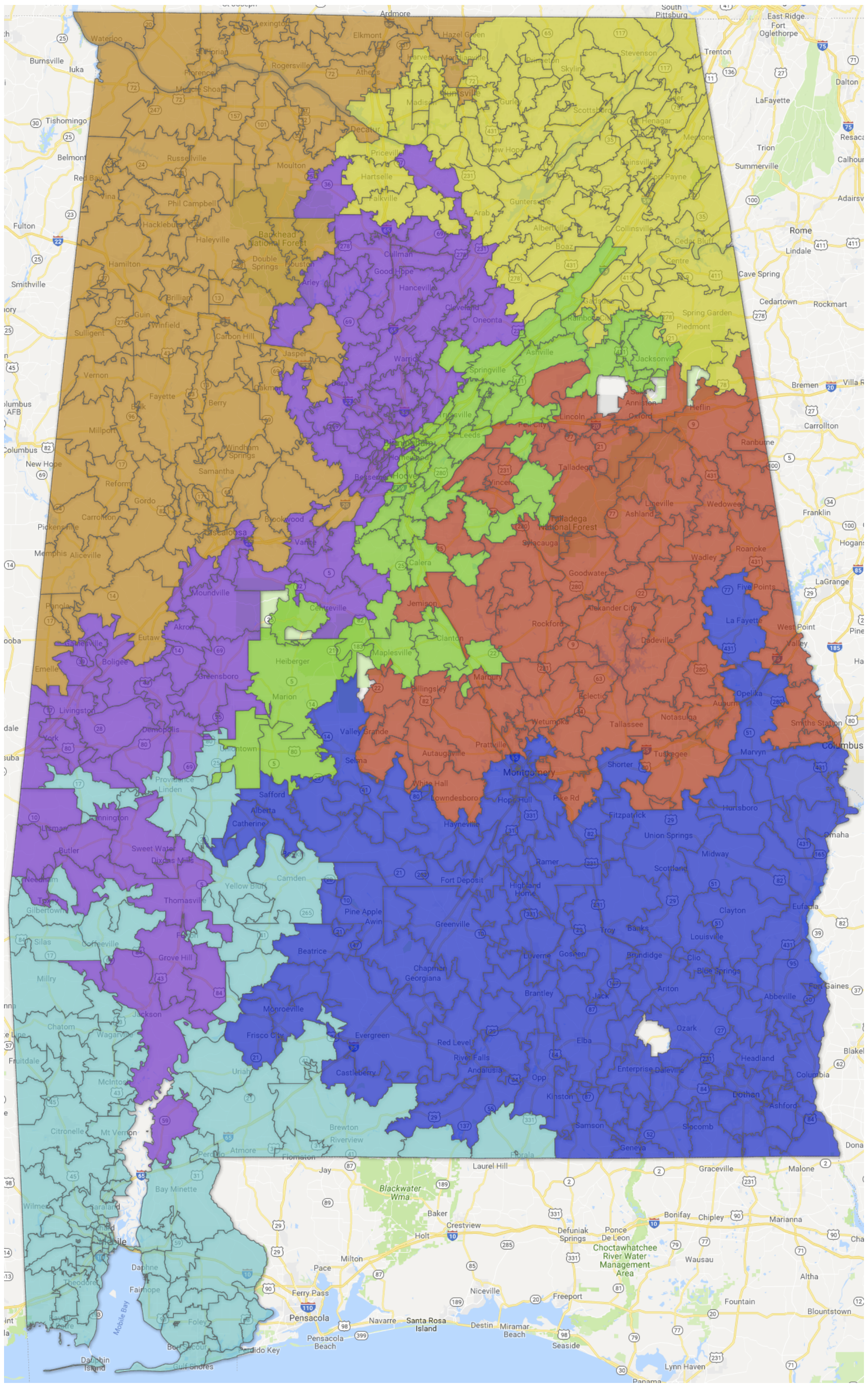

Figure 4.11: A sample near-optimal solution for districting the state of Alabama into seven contiguous zones comprised of ZIP Code Tabulation Areas (ZCTAs) based on population equality. The algorithm ran for 500 generations with a population size $\mu = 100$, neighborhood mutation with $p_m = 0.0025$, no recombination or crossover, a selection tournament size $k = 3$, and a soft contiguity constraint. The above solution represents a population inequality of at most 66 persons between all districts.

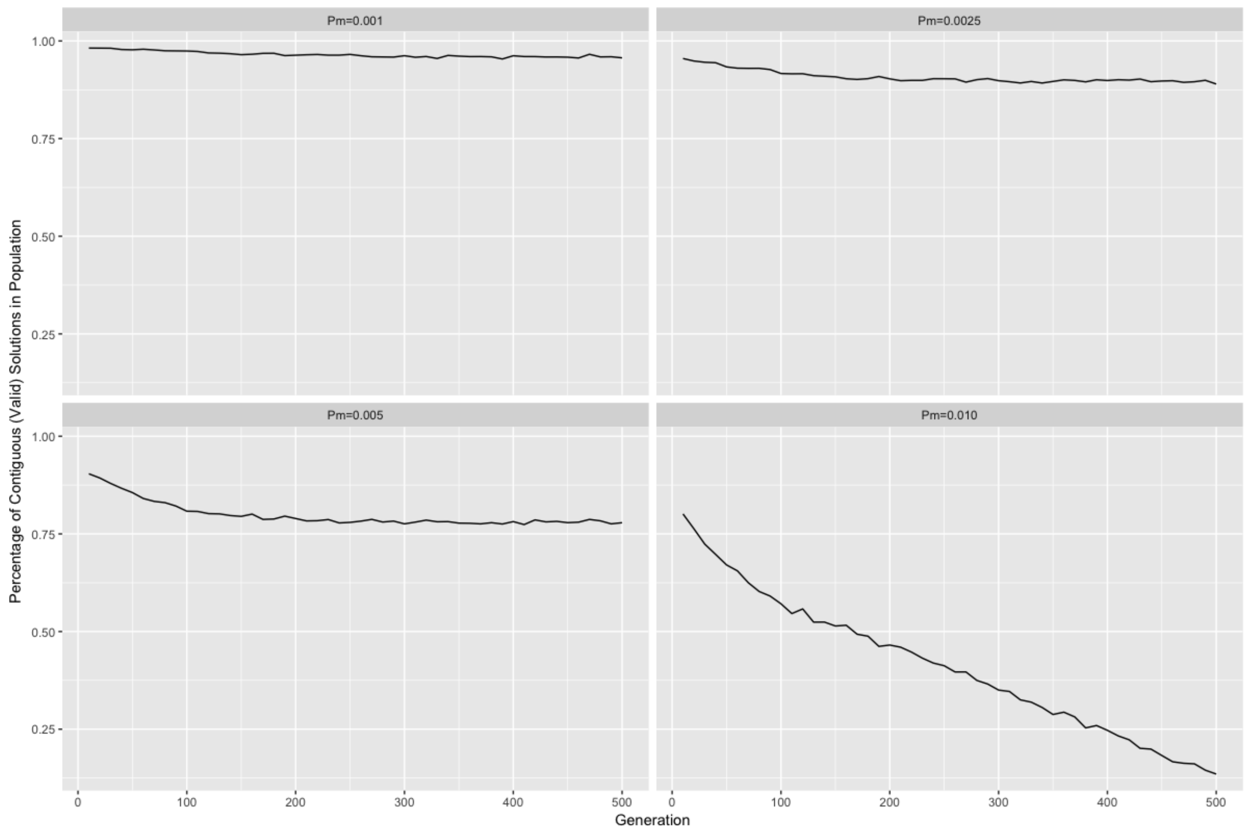

Figure 4.12: Percent of feasible (i.e. contiguous) solutions in the population, by generation, for 100 iterations of evolutionary search for solving a contiguous districting problem dividing the state of Alabama into seven districts. The algorithm employed a population size $\mu = 100$, no crossover or recombination, single-parent tournament selection with tournament size $k = 3$, and Neighborhood Mutation with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$).

The observation of poorer performance for the higher mutation rate is correlated with the increase in discontiguous, or infeasible, solutions that enter the population as the mutation rate increases. With higher mutation rates, such as $p_m = 0.010$, the ability of selective pressure to eliminate the occasional infeasible solution is overcome by the rate of introduction of new infeasible solutions, leading to a continuing decrease in the percentage of valid solutions in subsequent generations *(see Figure 4.12)*.

This can also be observed in the standard deviation of the population fitness over time, which shows a continued decrease and then stabilization at a relatively low level for the first

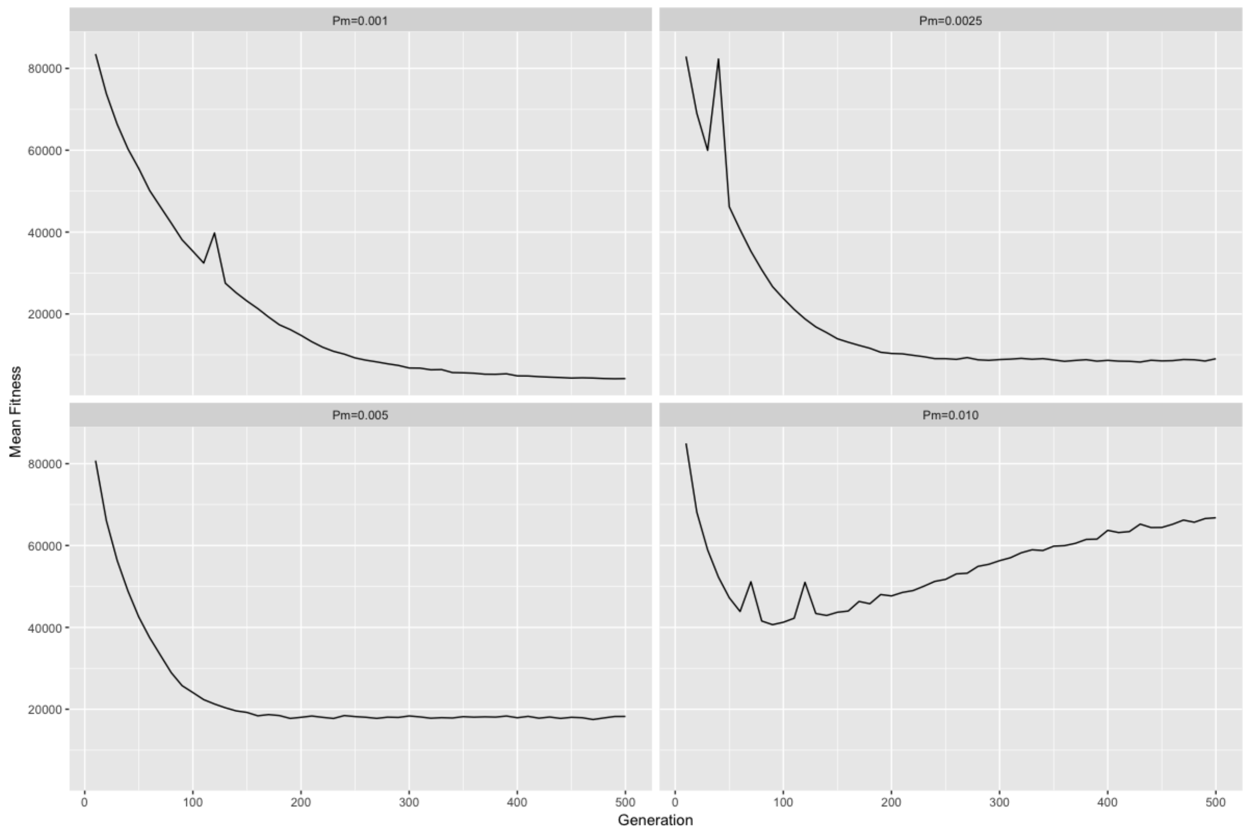

Figure 4.13: Mean fitness value, by generation, for 100 iterations of evolutionary search for solving a contiguous districting problem dividing the state of Alabama into seven districts. The algorithm employed a population size $\mu = 100$, no crossover or recombination, single-parent tournament selection with tournament size $k = 3$, and Neighborhood Mutation with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$).

three trials *(see Figure 4.15)*. However, for the fourth trial, we see that the standard deviation remains high over time, mainly due to the presence of a large percentage of infeasible solutions, which incur a significant fitness penalty and therefore widen the range between fit, valid solutions and these. Application of the Local Repair algorithm, in addition to Neighborhood Mutation, may prove of value in mitigating this effect, but was not explored because of the highly fit results already obtained with lower mutation rates.

This algorithm can be repeated with randomly assigned district centers rather than fixed centers to produce similar results in terms of population distribution but more variation in district shape *(see Figure A.2)*.

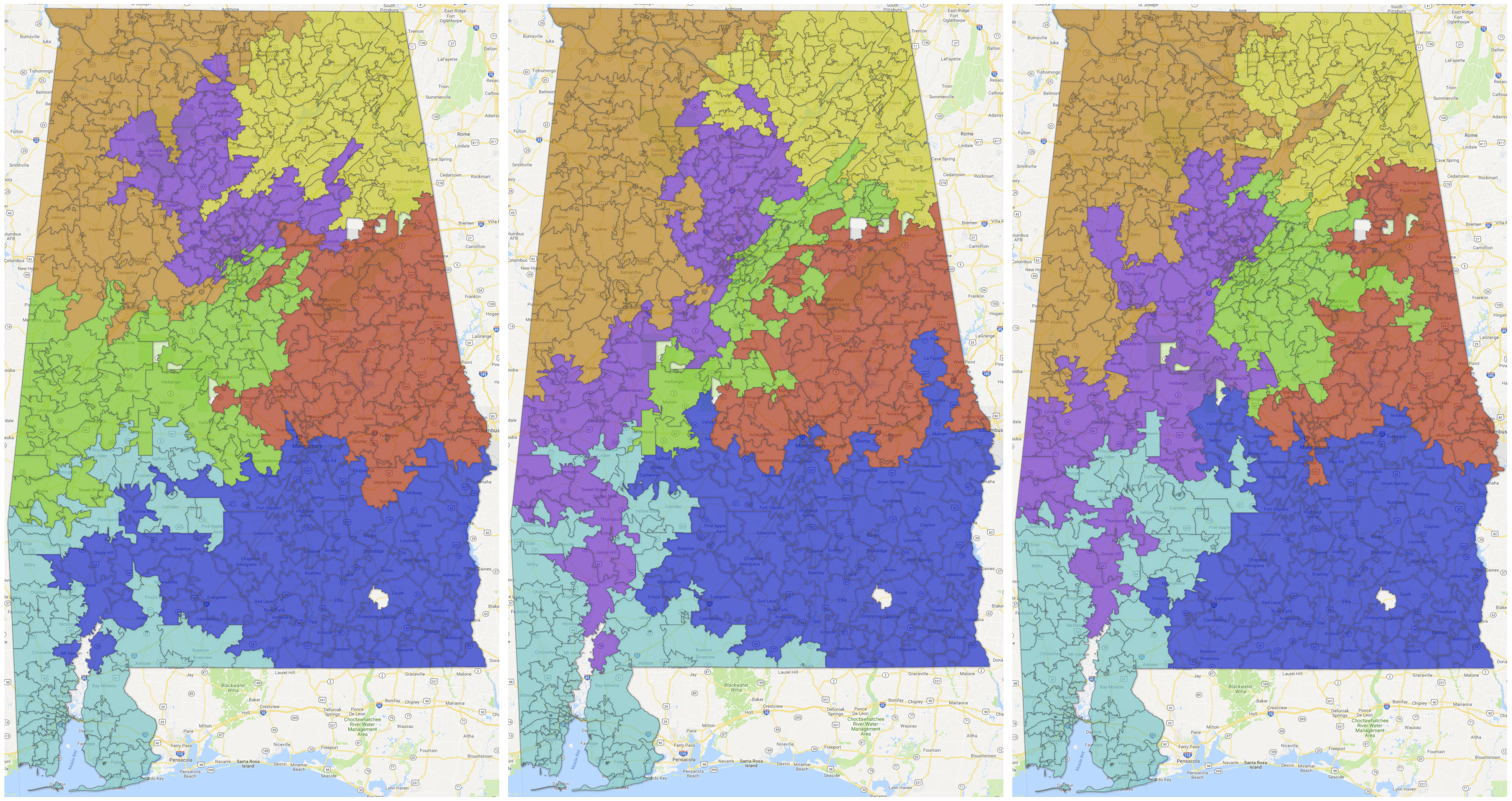

Figure 4.14: Three near-optimal solutions for districting the state of Alabama into seven contiguous districts based on population equality. Each solution represents the most-fit solution found after 100 iterations of an evolutionary algorithm that ran for 500 generations with a population size $\mu = 100$, no recombination or crossover, a selection tournament size $k = 3$, and a soft contiguity constraint. Neighborhood Mutation was used with varying mutation rates, from left, of $p_m = \{0.005, 0.0025, 0.001\}$. The solutions returned represent a population inequality, from left, of at most 78, 66, and 72 persons, respectively, between all districts.

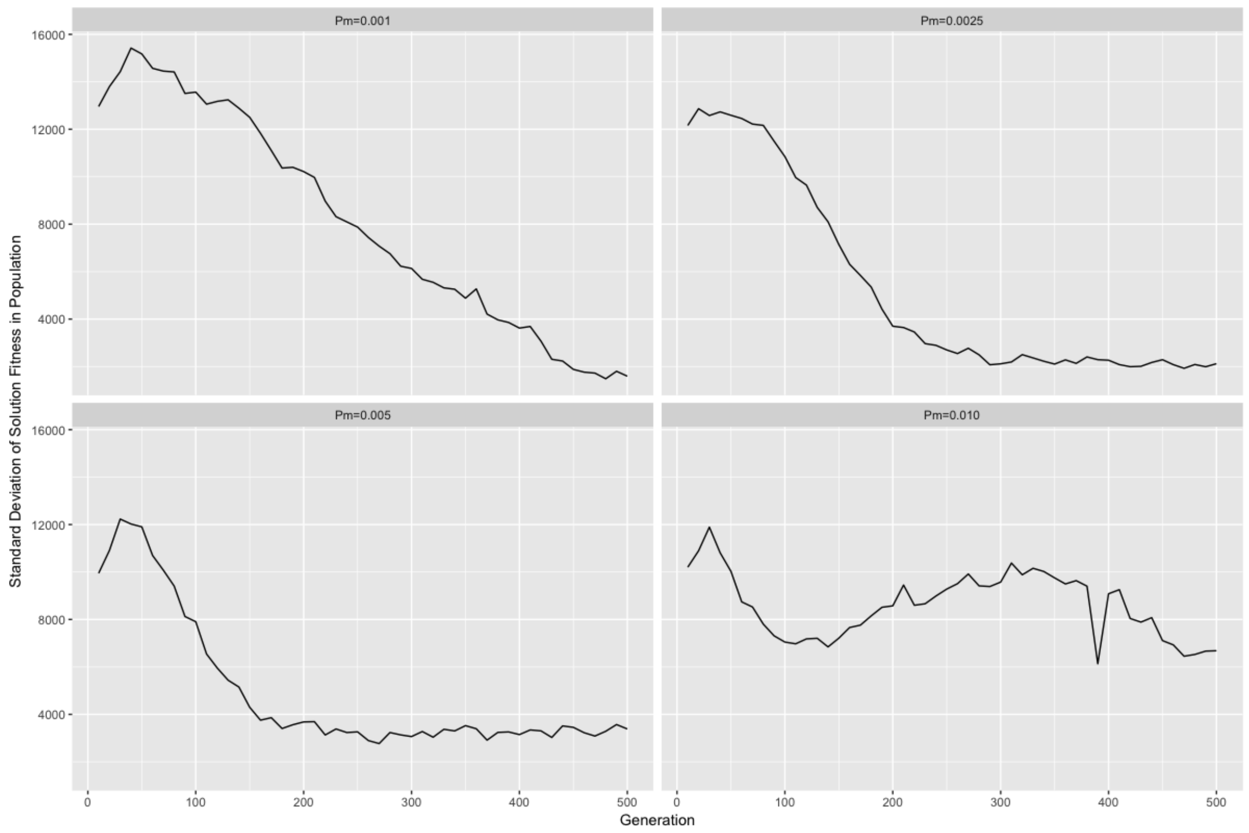

Figure 4.15: The standard deviation between fitness values in the population, by generation, for 100 iterations of evolutionary search for solving a contiguous districting problem dividing the state of Alabama into seven districts. The algorithm employed a population size $\mu = 100$, no crossover or recombination, single-parent tournament selection with tournament size $k = 3$, and Neighborhood Mutation with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$).

4.4.2 Performance on Minority Population Heuristic

The previous trials were repeated, this time with implementation of both the Neighborhood Mutation and the Local Repair algorithms, using an identical fitness heuristic method based on equal distribution of minority population between districts. The results show that the addition of the Local Repair algorithm to Neighborhood Mutation can be of benefit when mutation rates are increased *(see Tables A.8 and A.9)*.

The algorithm was able to produce extremely high quality solutions in all trials, with near-optimal solutions successfully districting the state of Alabama into seven districts based upon equality of minority population distribution with a deviation between any two districts

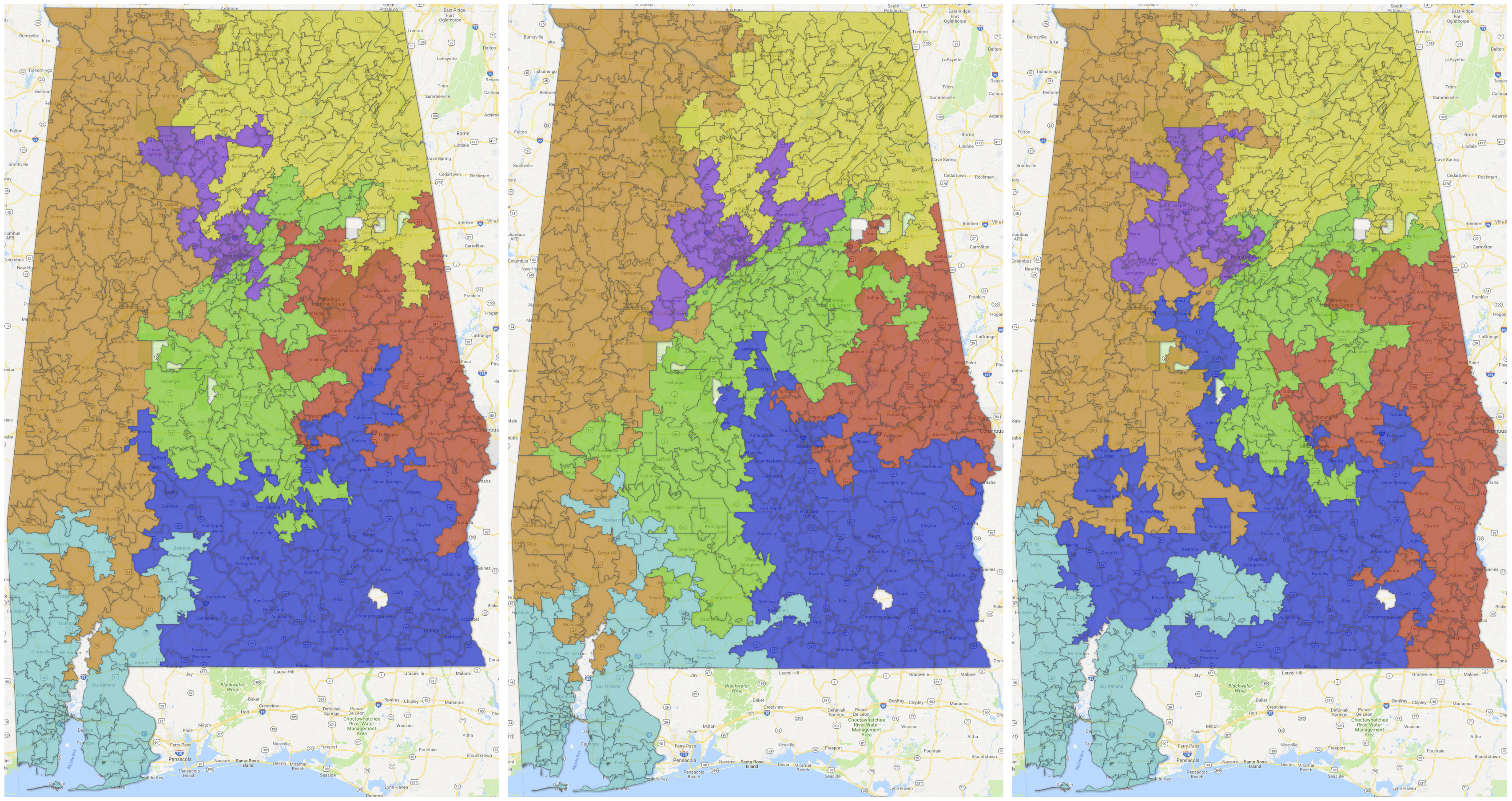

Figure 4.16: Three near-optimal solutions for districting the state of Alabama into seven contiguous districts based on minority population equality. Each solution represents the most-fit solution found after 100 iterations of an evolutionary algorithm that ran for 500 generations with a population size $\mu = 100$, no recombination or crossover, a selection tournament size $k = 3$, and a soft contiguity constraint. Neighborhood Mutation was used with varying mutation rates, from left, of $p_m = \{0.001, 0.0025, 0.005\}$. The solutions returned represent a population difference, from left, of at most 30, 10, and 17 minority persons, respectively, between any given districts.

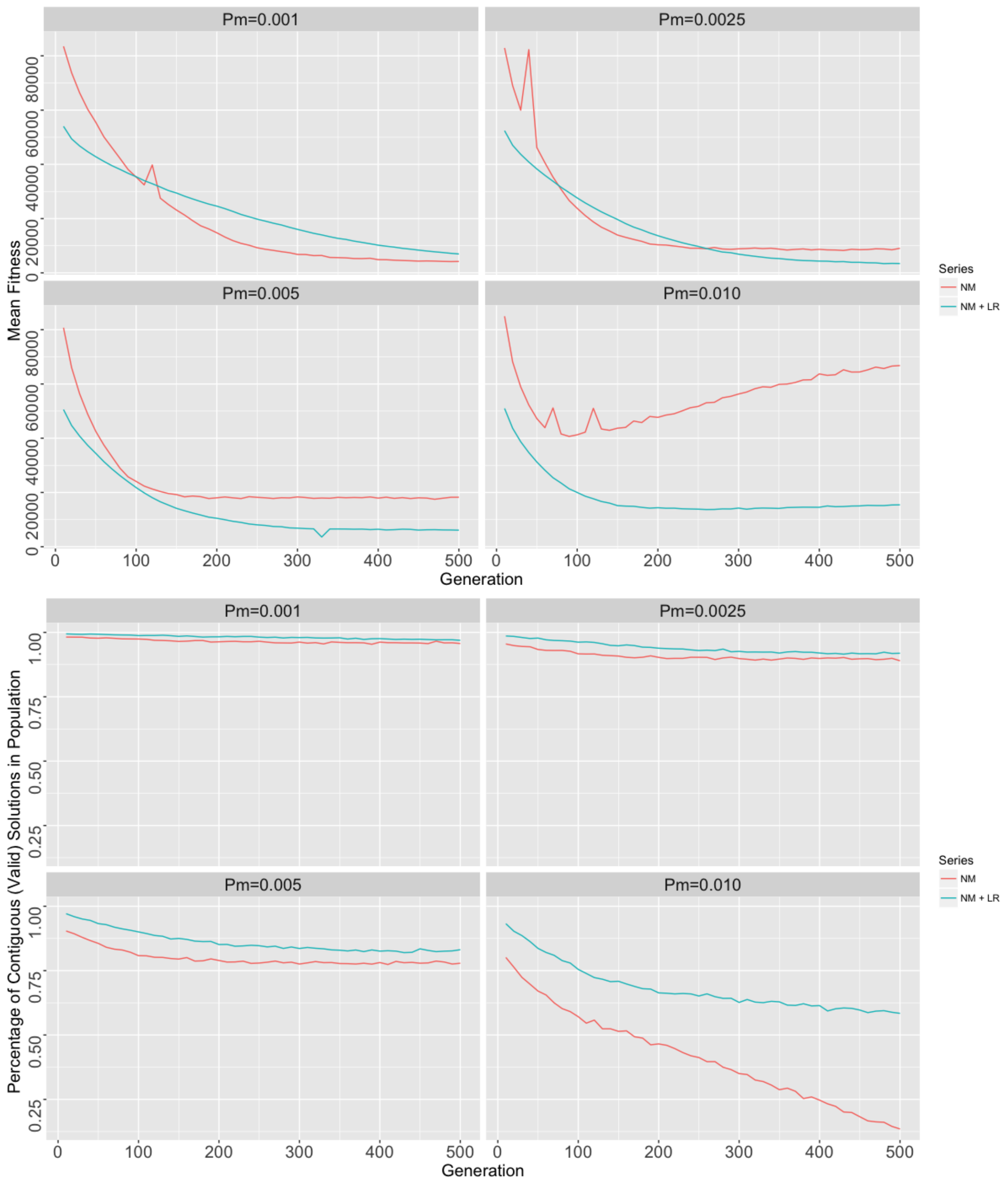

Figure 4.17: The mean fitness, top, and percentage of contiguous (i.e. valid) solutions, bottom, for all individuals in the population, by generation, for two series of a population equality heuristic with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$). The series indicated by the red line depicts an algorithm with Neighborhood Mutation only; the series with the blue line depicts a trial with Neighborhood Mutation combined with Local Repair.

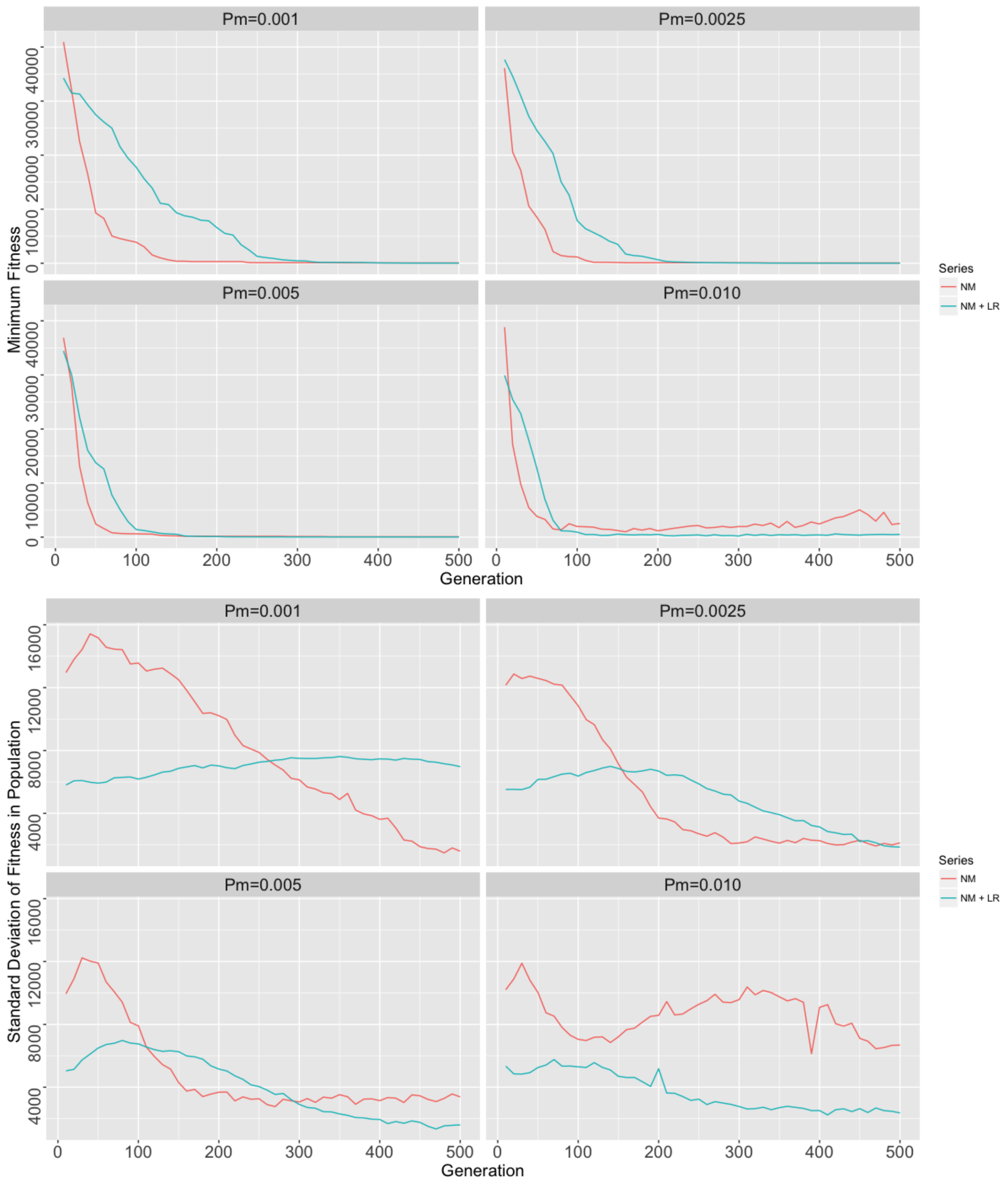

Figure 4.18: The minimum fitness, top, and fitness standard deviation, bottom, for all individuals in the population, by generation, for two series of a population equality heuristic with varying mutation rates over four trials ($p_m = \{0.001, 0.0025, 0.005, 0.010\}$). The series indicated by the red line depicts an algorithm with Neighborhood Mutation only; the series with the blue line depicts an algorithm with both Neighborhood Mutation and Local Repair.

of no more than 10 minority persons in the best case, and a deviation of 17 and 30 persons in two other cases *(see Figure 4.16)*.

The Local Repair algorithm is especially effective in mitigating the number of infeasible solutions that begin to dominate the population as mutation rates increase to a point where Neighborhood Mutation cannot prevent infeasible solutions from being generated faster than selection pressure can eliminate them. This rate has been observed to be about one in every 100 gene transcriptions, or $p_m = 0.010$. At or above this level, Local Repair is able to significantly increase the number of valid solutions in the population. Specifically, in trials with a mutation rate of $p_m = 0.010$, the percentage of contiguous, valid solutions in the population was observed to fall to as low as 13.51% in later generations with Neighborhood Mutation only. But with Neighborhood Mutation combined with Local Repair the percentage of contiguous, valid solutions in the populations never fell below 58.41% *(see Tables A.4 and A.9 and Figure 4.17)*. Similarly, the Local Repair algorithm was effective in curtailing the tendency of both the standard deviation of fitness and the minimum fitness value in the population to both increase over time for higher mutation rates *(see Figure 4.18)*.

4.4.3 Performance on District Compactness Heuristics

The algorithm was also evaluated with two district compactness heuristics, which are modeled on the distance minimization heuristics employed by Correa et al. in their solution of the P-Median problem [50] and Bação et al. [12]. In the first trial, district compactness is assessed as the sum of the distances from the centroid of each zone in the district to the centroid of that district center's own centroid. This is calculated using the half-versed sine, or haversine, formula for determining great-circle distance, which although it assumes a perfectly spherical Earth has negligible error for distances at this scale [32, 161-169]. The maximum of the sum of distances for each district, or the worst of the districts in the solution, is selected as the overall fitness for the entire solution. Specifically, this "sum of distances"

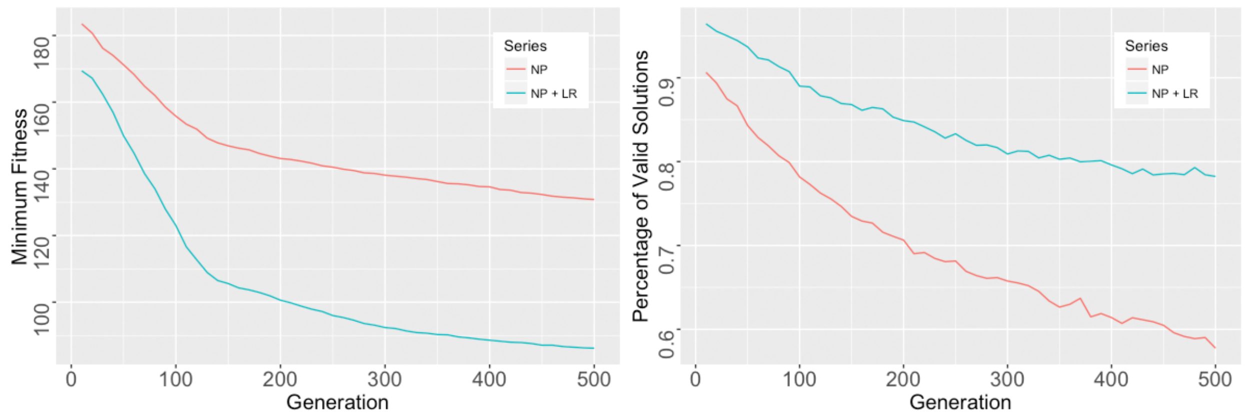

Figure 4.19: The minimum fitness, left, and percentage of contiguous (valid) solutions, right, for all individuals in the population, by generation, for 100 trials of a "sum of distances" district compactness heuristic for an evolutionary algorithms with population size $\mu = 100$, no recombination or crossover, single-parent selection with tournament size $k = 3$, and neighborhood mutation with rate $p_m = 0.005$. The series indicated by the red line depicts an algorithm with Neighborhood Mutation only; the series with the blue line depicts an algorithm with Neighborhood Mutation combined with Local Repair.

compactness heuristic can be expressed as follows:

$$\text{minimize} \quad \sum_{j=1}^{n}\sum_{i=1}^{n} d_{ij}x_{ij} \tag{4.2}$$

$$\text{subject to} \quad x_{ij} \in \{0,1\}, i,j = 1,2,...,n \tag{a}$$

where d_{ij} is the distance from the center of zone i to the center of district j and x_{ij} returns 1 iff zone i is assigned to district j, and 0 otherwise.

This heuristic produces reasonable, tightly clustered candidate solutions with minimum distances between zones *(see Table A.12)*, and a typical highly fit solution with maximum sum of distances from a district's zones to the district center of 8142.35 kilometers *(see Figure 4.20)*.

We again see that the Local Repair algorithm is especially effective in both increasing the quality of the solution returned and also the percentage of contiguous (i.e. valid) solutions in the population over time *(see Table A.12)*. We can observe that, with Neighborhood Mutation only, the percentage of valid solutions decreases to as little as 57.75% with a mutation rate of $p_m = 0.005$. However, with the addition of the Local Repair algorithm, that percentage remains at or above 78.23% for the duration of the algorithm. Similarly, the minimum fitness solution found is consistently and significantly lower for the duration of the algorithm's run *(see Figure 4.19)*.

A second variation attempts to minimize the mean distance from the centroid of each zone in the district to the district center's own centroid, and can be expressed as follows:

$$\text{minimize} \quad \sum_j \left(\frac{\sum_{i=1}^n d_{ij} x_{ij}}{\sum_{i=1}^n x_{ij}} \right) \tag{4.3}$$

$$\text{subject to} \quad x_{ij} \in \{0,1\}, i, j = 1, 2, ..., n \tag{a}$$

where d_{ij} is the distance from the center of zone i to the center of district j and x_{ij} returns 1 iff zone i is assigned to district j, , and 0 otherwise.

This heuristic can produce slightly more elongated districts, as seen in Figure 4.21, which represents a typical highly fit solution with maximum average distance from a district's zones to the district center of 93.2671 kilometers *(see also Table A.11)*.

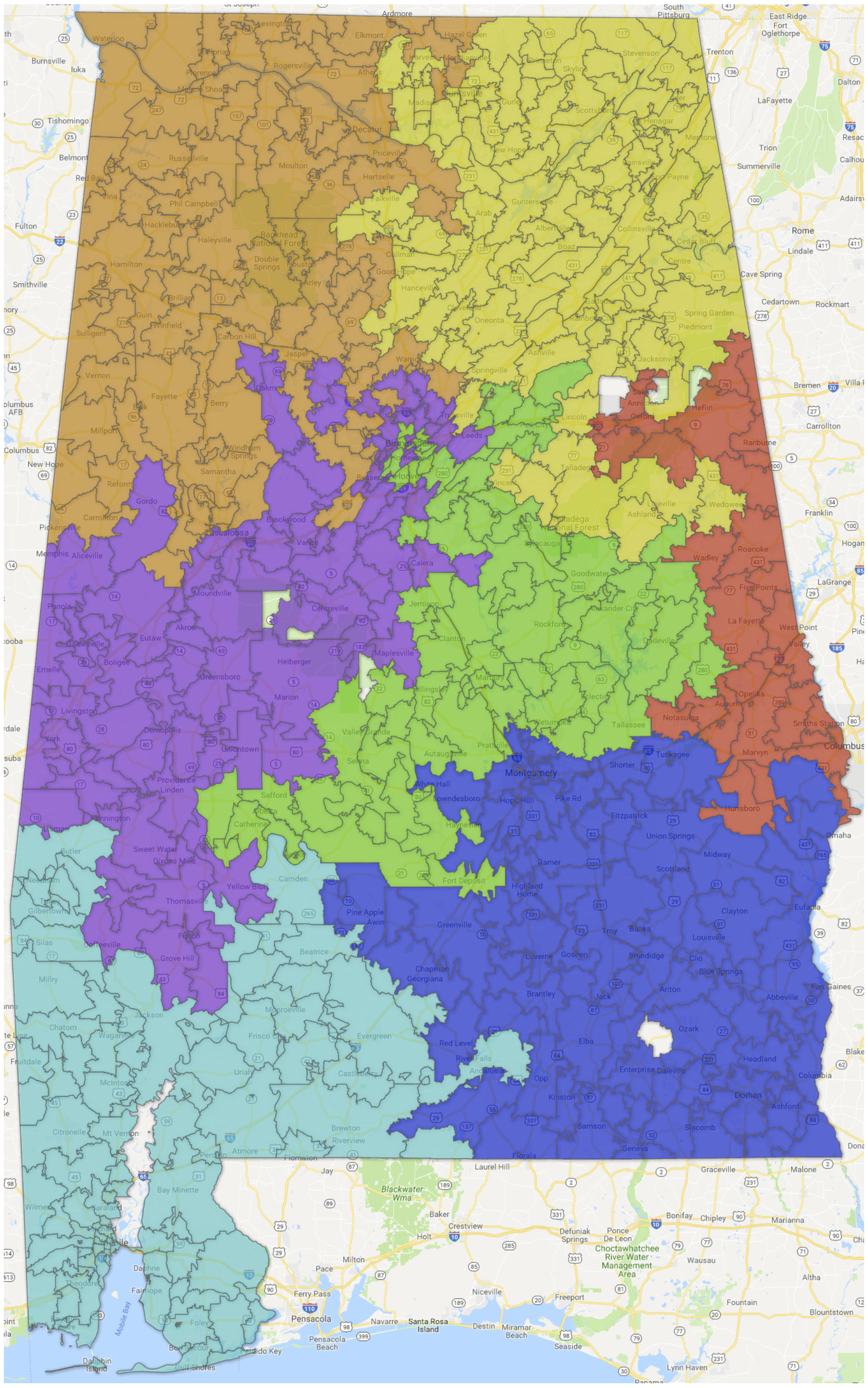

Figure 4.20: An alternative solution to a districting problem for the State of Alabama using a district compactness heuristic based upon minimizing the sum of the distances of zone centroids to their respective district centers' centroid.

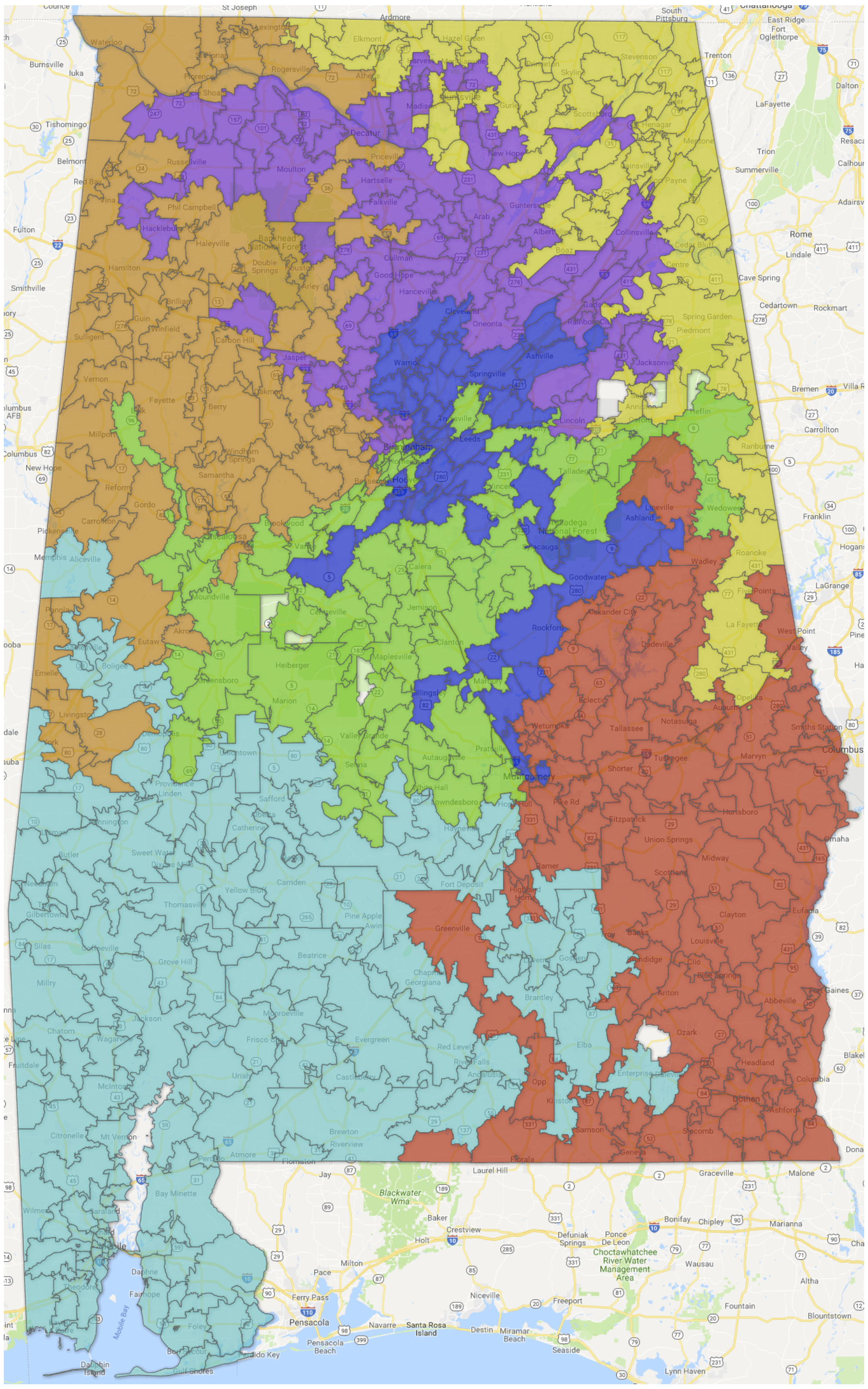

Figure 4.21: A sample near-optimal solution for districting the state of Alabama into seven contiguous zones comprised of ZIP Code Tabulation Areas (ZCTAs) based on district compactness, as measured by the average distance from each parcel's centroid to its district center's centroid. The algorithm ran for 500 generations with a population size $\mu = 100$, neighborhood mutation with $p_m = 0.005$, no recombination or crossover, a selection tournament size $k = 3$, and a soft contiguity constraint. The above solution represents a maximum average distance to the district center of 93.2671 km.

Chapter 5
Evolutionary Algorithms in Multi-criteria Optimization Search and the Use of Agents in Heuristic Evaluation

"*If we are to understand the interactions of a large number of agents, we must first be able to describe the capabilities of individual agents.*"

— John Henry Holland [110]

5.1 Implementing a Multi-Objective Heuristic for Districting Problems

Most real-world search problems can seldom be reduced to a single heuristic. More often, there are multiple, sometimes conflicting, considerations that must be taken into account in order to produce an acceptable solution. Konak et al. write that, when it comes to multi-objective search problems, "the objectives are generally conflicting, preventing simultaneous optimization of each objective" and that "many, or even most, real engineering problems actually do have multiple objectives" [107].

Geographic districting problems are no different, and examples can easily be envisioned. For example, a warehousing problem might need to reach a balance between delivery times and distances with fuel consumed and labor hours paid for drivers. Or, an emergency medical service's dispatching plan might have to balance the location of basic or advanced life support ambulances with average response times and the location of supporting fire departments and rescue services.

Rittel and Webber call these "wicked problems" because they have few clarifying traits and "include nearly all public policy issues — whether the question concerns the location of a freeway, the adjustment of a tax rate, the modification a school curricula, or the confrontation of crime" [147, 160]. They write that the information needed to solve such problems depends on each particular stakeholder's interpretation or approach for solving it. When dealing with such real-world problems, Xiao et al. similarly write that "decision makers may find it difficult to derive a single solution that is best in all objectives" [185].

Single-objective optimization problems are relatively straightforward, and finding a solution that satisfies or nearly satisfies a single criteria is relatively trivial. Traditional and evolutionary search algorithms could easily tackle such problems if only one constraint were addressed at a time. But near-optimal solutions for one constraint may be very poorly suited from the perspective of another constraint.

There are many approaches to find solutions in a mutli-objective problem. Among these are the constraint method, the weighted mean or [134]"scalarization" methods [157], and the noninferior set estimation (NISE) method [48], which is a particular application of a weighted method that attempts to derive an optimal set of weights for individual criteria.

With the constraint method all but a single, primary objective is moved to a constraint set, in which the range of each constraint criteria is allowed to vary freely within a defined range of allowed tolerance [47][107].

But the more commonly employed of these methods is combining multiple objectives into a single heuristic function, but with this approach "the problem lies in the proper selection of the weights or utility functions to characterize the decision maker's preferences," and it can be challenging to determine the proper weights for each individual objective, even for an expert in the subject domain [107].

When utilizing a combined heuristic with scaling of multiple objectives, a complication is that even small perturbations in weights of individual criteria can lead to vastly different solutions. In these cases, a “reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objective at an acceptable level without being dominated by any other solution” [107].

This method suffers from drawbacks: it is useful only for problems that can be expressed mathematically; it is inefficient when applied to large problems; and it may fail to find important solutions [125][185]

When a multi-objective solution satisfies each objective as best it can, and in such a way that any improvement in regard to one criteria would result in a corresponding sacrifice in regard to one or more other criteria, that solution is said to be “nondominated”. Such a solution is said to be part of the Pareto optimal set, or the set of all solutions that are nondominated. There are many, and possibly an infinite, number of Pareto optimal solutions for a multi-objective problem, and the set of these solutions can be envisioned to form a line in the n-dimensional hyperspace that is called the “Pareto front” or “Pareto frontier” [47][107][132].

In terms of multi-objective search, this means that when two or more objectives conflict with each other, “optimizing (a heuristic) with respect to a single objective often results in unacceptable results with respect to the other objectives” and therefore “a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible” [107].

There are three general goals for a multi-objective optimization problem: [47][107]

1. The best-known Pareto front should be as close as possible to the true Pareto front. That is, the algorithm should endeavor to find a best solution or solutions that are as close to the theoretically optimal set as possible.

2. Solutions in the best-known Pareto set should be evenly distributed over the frontier in order to provide the user a clear understanding of tradeoffs between different solutions.

3. The best known Pareto front should represent the spectrum of the entire front, which requires investigating solutions at the extreme ends of the search space.

Evolutionary Algorithms in general, and Genetic Algorithms in particular, are ideal for exploration of such a frontier, and are by far the method of choice for multi-objective metaheuristic techniques [99][185].

Political districting problems are an excellent example of a problem with multiple, competing constraints that must be balanced: population equality, compactness, preservation of neighborhoods and communities, distribution or consolidation of minority or ethnic populations, household income, home ownership, respect for natural boundaries, etc.

Finding a solution that satisfies all, or at least many, of these constraints is extremely difficult, and in the end comes down to a subjective decision made by a stakeholder. But it is still necessary to employ search to get to a point at which a set of one or more potential solutions that balance these considerations can be compared subjectively by the user. For this reason, it is necessary to find a way to combine multiple, independent heuristics into a single objective function that can be utilized by search algorithms, whether traditional or evolutionary, to order the desirability of potential solutions with respect to each other.

The approach of combining independent heuristics into a single objective function, as employed by Bozkaya et al., is most commonly accomplished by treating each criteria as a weighted constraint with an independent weighting factor [29][107]. These weighting factors are either arbitrarily assigned by the user or may be allowed to fluctuate over the course of the algorithm's execution.

In our implementation of a weighted multi-objective objective function, we first identify the independent heuristics that we wish to evaluate. For example, we seek a solution for a districting problem that balances the following considerations: population equality between districts; equal distribution of minority population between districts; geographic compactness of districts; and geographic contiguity (i.e. "connectedness") of all zones in a district.

Both population equality heuristics measure the divergence of each zone's population with the average population for all zones. The difference between the average and the value of the zone with the maximum divergence is then selected as the overall fitness of the solution — essentially, the zone with the worst difference from the average drives the fitness.

Therefore, the heuristic for total population equality can be expressed as follows:

$$f_{pop}(x) = \max_{j \in J} \left(\left| P(j) - \bar{P} \right| \right) \tag{5.1}$$

$$\text{subject to} \sum_{j \in J} x_{ij} = 1 \quad (i \in I), \tag{a}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i \in I, j \in J), \tag{b}$$

$$\text{where} P(j) = \sum_{i \in I} p_i x_{ij} \quad (j \in J), \tag{c}$$

$$\bar{P} = \frac{\sum_{j \in J} \sum_{i \in I} p_i x_{ij}}{k} \tag{d}$$

where $P(j)$ is the sum of the population in district j indicated by (c), $\bar{P}$ is the average of the populations of all districts indicated by (d), x_{ij} in (a) is a constraint function that each zone is assigned to one and only one district, and (b) is a constraint function that is equal to one iff zone i is assigned to district j.

To equalize the distribution of minority population, the heuristic is identical to the above formula with the exception of the population data being used:

$$f_{minority}(x) = \max_{j \in J} \left(\left| P_{minority}(j) - \bar{P}_{minority} \right| \right) \tag{5.2}$$

with the same constraints as above.

For measuring district compactness, we have the two heuristics proposed by Bozkaya et al. and the two employed here in previous trials. The more subjectively appealing of these is the sum-of-distances measure used in the previous chapter *(see Figure 4.20)*, but the average of distances and other measures are equally valid. Therefore, the district compactness measure can be expressed as follows:

$$f_{comp}(x) = \sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} x_{ij} \tag{5.3}$$

$$\text{subject to} \quad x_{ij} \in \{0,1\}, i, j = 1, 2, ..., n \tag{a}$$

where d_{ij} is the distance from the center of zone i to the center of district j and x_{ij} returns 1 iff zone i is assigned to district j,

Combining these three measures into a single objective function requires a separate weighting measure for each, which we will call α_{pop}, $\alpha_{minority}$, and α_{comp}. The combined objective function can then be expressed as the sum of each individual fitness heuristic multiplied by its weighting measure:

$$F(x) = \alpha_{pop} f_{pop}(x) + \alpha_{minority} f_{minority}(x) + \alpha_{comp} f_{comp}(x) \tag{5.4}$$

To accommodate the soft contiguity constraint, we forego the fixed-value fitness penalization employed previously in favor of a proportionate penalty that multiplies the fitness value by a user-defined scaling factor. First, this approach maintains the advantage of retaining invalid but otherwise highly fit solutions in the population, as advocated by Bação et al., Chou et al., and others [12][44]. But instead of clustering all infeasible solutions at the bottom of a fitness-ordered population, where selective pressure would quickly eliminate them, a proportionate penalty allows the infeasible solutions to be distributed more evenly. In this manner, a highly fit but infeasible solution, even after penalization, could have a better fitness value than a valid but poorly fit solution. Therefore, the highly fit but invalid solution has a greater chance of selection and further mutation that could, potentially, render the solution valid once more.

In this manner, the fitness function becomes:

$$F(x) = \beta\Big(\alpha_{pop} f_{pop}(x) + \alpha_{minority} f_{minority}(x) + \alpha_{comp} f_{comp}(x)\Big) \tag{5.5}$$

where β is a scaling factor that is equal to 1 if the solution is contiguous (i.e. valid), and is equal to an arbitrary value greater than 1 when the solution is discontiguous. In this manner, β enforces the soft contiguity constraint by penalizing the fitness value, increasing it by a desired factor.

5.2 Evolutionary Search Results with a Multi-objective Heuristic

With this objective function, we evaluate the evolutionary algorithm's performance on a state-wide scale, districting the state of Alabama into seven contiguous zones based on this balance of three fitness heuristics and a soft contiguity constraint.

Scaling factors for the individual fitness function are chosen subjectively and vary depending on the which of the objective criteria are employed. We evaluate the performance of the Evolutionary Algorithm with four different combinations of objectives:

- Balancing total population distribution and district compactness with a soft contiguity constraint.
- Balancing minority population distribution and district compactness with a soft contiguity constraint.
- Balancing total population distribution and minority population distribution with a soft contiguity constraint.
- Balancing total population distribution, minority population distribution, and district compactness with a soft contiguity constraint.

Because the values of the first two criteria — total population distribution and minority population distribution — approach a theoretical optimal value of zero, the values of α_{pop}

Subjective Criteria Weights for Multi-Objective Optimization, By Trial				
	α_{pop}	$\alpha_{minority}$	α_{comp}	β
Total Population Distribution + District Compactness	0.005	0.00	1.00	2.5
Minority Population Distribution + District Compactness	0.00	0.05	1.00	1.5
Total Population Distribution + Minority Population Distribution	0.01	1.00	0.00	1.5
Total Population Distribution + Minority Population Distribution + District Compactness	0.01	1.00	1.00	1.5

Table 5.1: Subjective weighting factors for four trials of a weighted, multi-heuristic districting problem. The weights shown here correspond to the objective function depicted in Equation 5.5.

and $\alpha_{minority}$ tend to dominate the objective function compared to the district compactness measure. In terms of the two population distribution criteria, the total population is much larger and therefore tends to dominate the minority population distribution.

The specific criteria weights for each of these trials are depicted in Table 5.1. These weights are subjectively selected to scale each independent heuristic to an essentially equal weight, in terms of selective pressure and convergence of the algorithm.

The first two trials each utilize an heuristic function that attempts to balance three different objective simultaneously: equal distribution of population, district compactness, and district contiguity. In the first trial, a distribution of total population metric is used. In the second, a distribution of non-white, or minority, population metric is used. The two metrics are computed identically, with the only difference being the quantity of persons to divide equally. These three objectives — population distribution, compactness, and contiguity — are considered to be relatively complementary, but which we mean that with proper weighting factors no one objective tends to excessively dominate another.

Although there are several methods, particularly NISE, that can provide a set of weights that produce solutions along the Pareto front, that is not necessary here, as we desire only to provide sufficiently reasonable multi-objective solutions against which to later compare the performance of an agent-based solution.

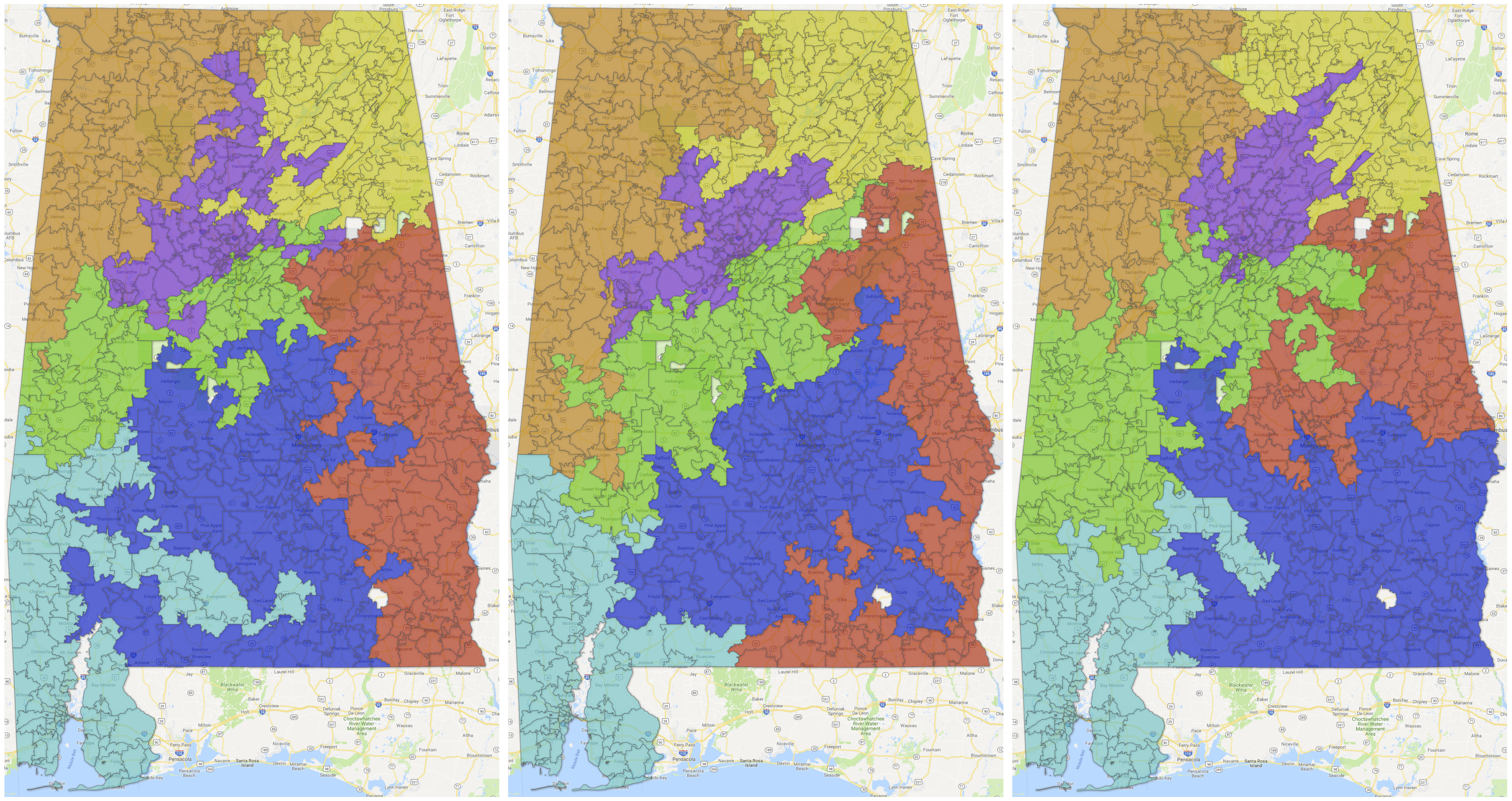

Figure 5.1: Three possible solutions for districting the state of Alabama into seven contiguous districts based on a multi-criteria heuristic that includes equal distribution of total population, district compactness, and geographic contiguity. The weighted heuristic attempts to achieve balance between the first two criteria and includes a penalization scaling factor for district discontiguity. The solutions represent, from left, a maximum total population difference between districts of 207, 329, and 205 persons. Compactness, measured as the maximum of the average distance of zones from the district center, is, from left, 321.4km, 378.7km, and 388.8km.

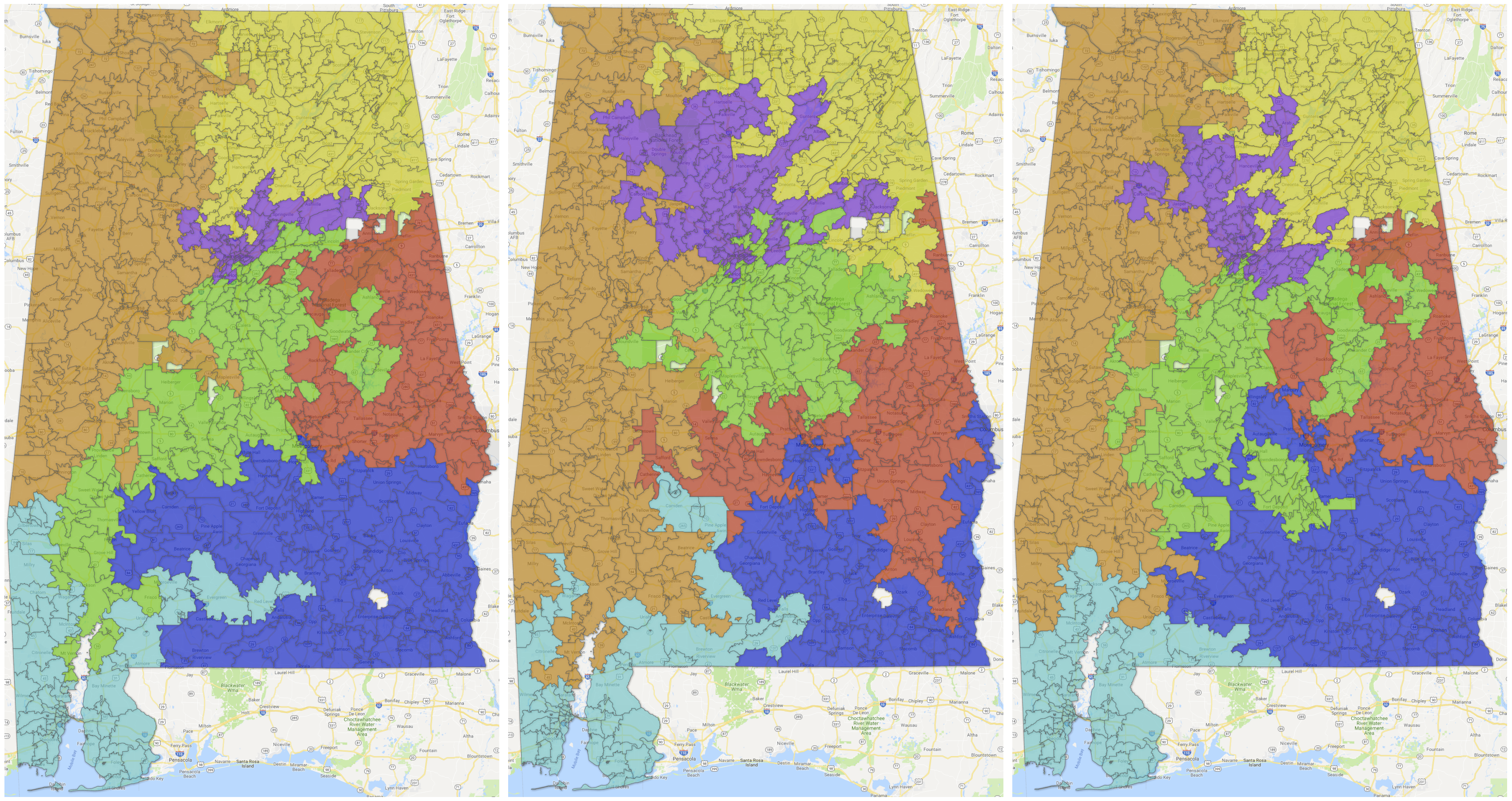

Figure 5.2: Three possible solutions for districting the state of Alabama into seven contiguous districts based on a multi-criteria heuristic that includes equal distribution of minority population, district compactness, and geographic contiguity. The weighted heuristic attempts to achieve balance between the first two criteria and includes a penalization scaling factor for district discontiguity. The solutions represent, from left, a maximum minority population difference between districts of 127, 161, and 170 persons. Compactness, measured as the maximum of the average distance of zones from the district center, is, from left, 336.2km, 374.2km, and 354.41km.

Algorithm performance in for the first trial shows similar performance to single-objective algorithm performance. Three solutions with subjectively reasonable results are shown in Figure 5.1 and represent successful partitioning of the state into seven districts with a difference of no more than 207, 329, and 205 persons, respectively, with reasonable compactness metrics. All four series of the algorithm, each with varying mutation rates of $p_m = 0.001, 0.0025, 0.005, 0.010$, show a reasonable downward curve for population mean fitness over successive generations. Similarly, the standard deviation of population fitness continues to grow tighter for all but the final series with a high mutation rate ($p_m = 0.010$) *(see Tables A.13 and A.14)*.

Performance for the second trial, which utilizes a distribution of minority population heuristic, is substantially similar, with three sample districting solutions that represent a difference of no more than 127, 161, 170 non-white persons with acceptable district compactness measures *(see Figure 5.2 and Tables A.15 and A.16)*.

Finally, two trials were run utilizing a multi-objective heuristic with two criteria that are in conflict with each other: distribution of total population and distribution of non-white, or minority population. A potential solution that tends to favor either one of these objectives tends to be dominant over the other objective. Although, with the correct weighting it is theoretically possible to achieve a candidate solution in which no objective dominates over the others — such a solution would be said to lie along the Pareto frontier for this problem.

The third trial balances only the total population and minority population distribution with a soft contiguity constraint and achieves only moderate success. Three solutions depicted in Figure 5.3 show substantially poorer distribution of each separate population while attempting to reach a balance of the two together. For example, with the weights used here, the total population difference between any two zones in candidate solutions is typically around 85,000 persons while the maximum difference in non-white persons between any two zones is around 1,500 persons. Adjustment of weights can shift this balance toward or away from one objective — essentially adjusting its dominance relative to the other. But, unless

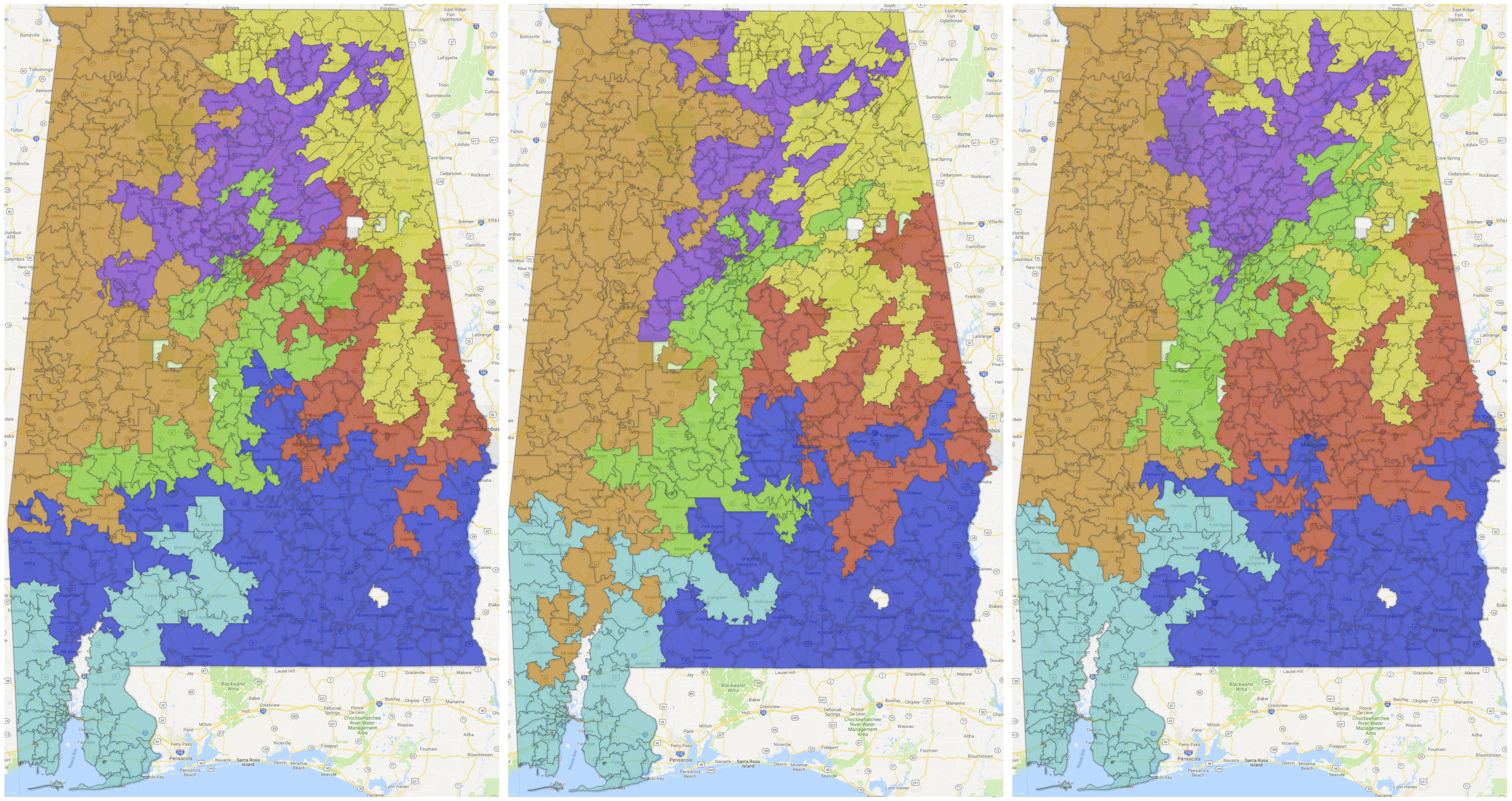

Figure 5.3: Three possible solutions for districting the state of Alabama into seven contiguous districts based on a multi-criteria heuristic that attempts to balance total population distribution, minority population distribution, and geographic contiguity. The weighted heuristic attempts to achieve balance between the first two criteria and includes a penalization scaling factor for district discontiguity. The solutions represent, from left, a maximum total population difference between districts, from left, of 85,969, 104,269, and 87,525 persons. Maximum minority population difference between districts, from left, is 1,474, 1,275, and 11,509 persons.

the weight of one objectives is reduced to near-zero, the results achieved for either objective will not be as precise as those that can be achieved with that objective in isolation.

Duplication of this with the addition of a fourth objective — a district compactness constraint — produces similar results. Two candidate solutions for a four-objective heuristic are shown in Figure 5.4.

When employing a weighted multi-heuristic function with conflicting objectives, we see that the standard deviation of the population fitness no longer grows continually tighter in successive generations, as seen in almost all other trials of this algorithm *(see Tables A.17, A.18, A.19, and A.20)*, confirming the inability of the algorithm to continually improve the population in each successive generation.

These trials can be repeated with randomly assigned district centers rather than fixed centers, which tend to produce slightly better results, especially in terms of the district compactness objective.

5.3 Multi-objective Heuristic Evaluation by Independent Agents

Although its been more than two decades since the idea of intelligent agents became mainstream, there still lacks consensus on a single definition for the term "agent" [182] [96] [139] [116] [78]. When it comes to describing what constitutes an intelligent software agent, Jennings writes that "there is no real agreement even on the core question of exactly what an agent is," much less what comprises one category of agents versus another [96, 8]. Heucke, Wooldridge, and others agree, noting that to attempt their own definitions would be, as Gilbert et al. write, to add to an "already crowded plate" [78] [87, 1598]. Wooldridge, with Jennings, suggests that the entire field of artificial intelligence can be summarized as that part of computer science that "aims to construct agents that exhibit aspects of intelligent behavior" and that the idea of an "agent" is therefore a central concept of artificial intelligence [182, 116]. Wooldridge elects to punt on attempting a formal description, concluding that the term agent "defies attempts to produce a single universally accepted definition"

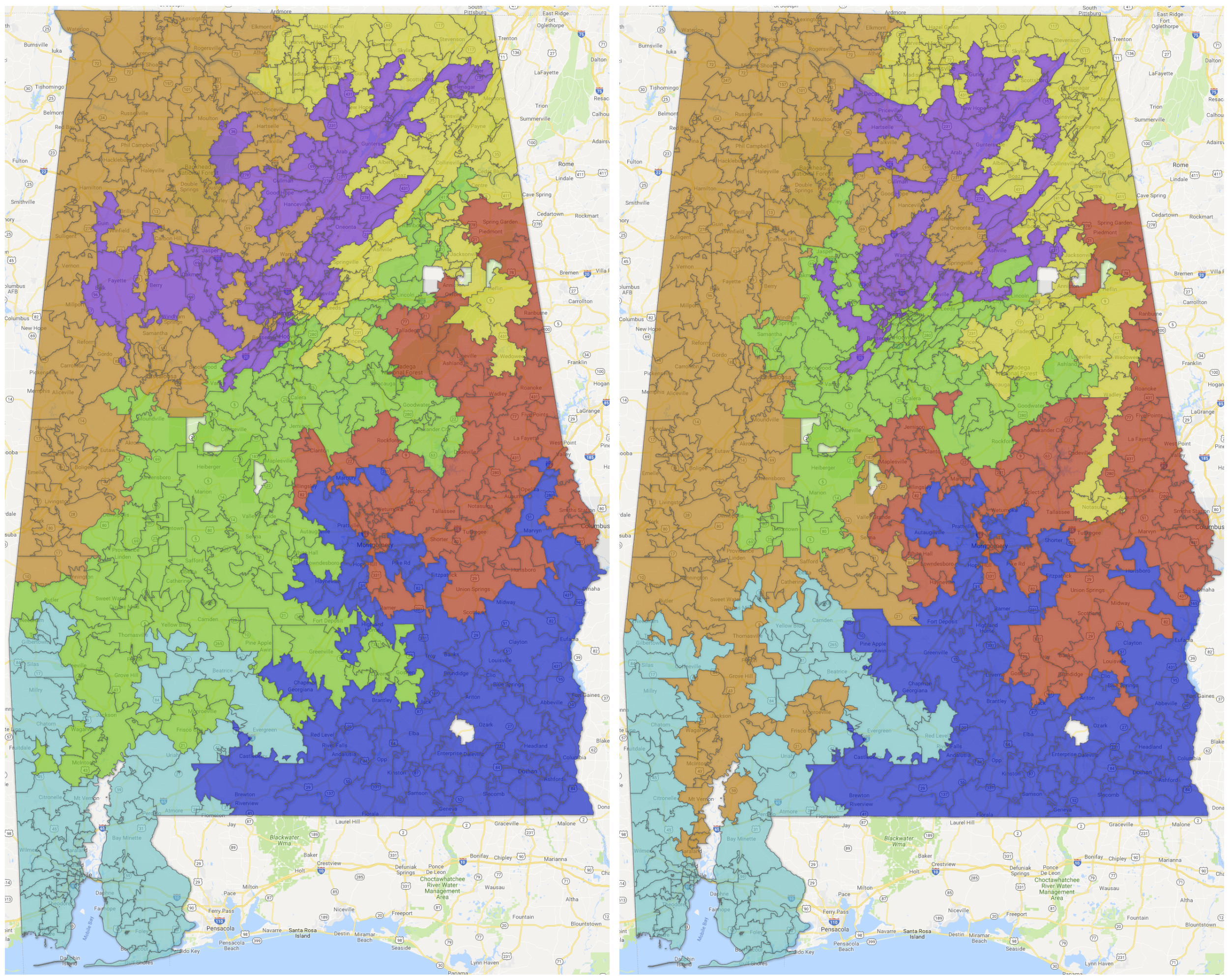

Figure 5.4: Two possible solutions for districting the state of Alabama into seven contiguous districts based on a multi-criteria heuristic that attempts to balance total population distribution, minority population distribution, district compactness, and geographic contiguity. The weighted heuristic attempts to achieve balance between the first three criteria and includes a penalization scaling factor for district discontiguity. The solutions represent, from left, a maximum total population difference between districts of 112,783 and 108,659 persons, maximum minority population difference of 185 and 864 persons, and a maximum average distance from the district center of 321.3km and 443.91km.

[182, 116]. Nwana writes that overuse of the word has tended to "mask the fact that...there is a truly heterogeneous body of research being carried out under this banner" [133].

A universal definition may be impossible, but most scholars seem to be in general agreement regarding many of the characteristics that software agents share, although they often differ in terminology and perspective.

Russell and Norvig suggest that the most important characteristics are an agent's ability to perceive its environment and to effect actions upon that environment [153]. Jennings et al. agree, and call this *situatedness*, which "means that the agent receives sensory input from its environment and that it can perform actions which change the environment in some way" [96, 8].

Franklin and Graesser expand on this with the addition that the agent's actions reflect pursuit of an agenda specific to the agent [72]. Heucke and Gilbert et al. extrapolate this further by making the agent's agenda a proxy for that of a human operator or user and write that agents "...employ some knowledge or representation of the user's goals or desire" [87, 7-38] [78, 913].

Wooldridge and Jennings enumerate characteristics likely to be found in software agents: autonomy, social ability, reactivity, and pro-activeness [182]. Gilbert et al. define three axes of intelligent agents: Agency, Intelligence, and Mobility [78, 913]. Heucke, based on an earlier classification by Nwana, also attempts a taxonomy of software agents, which includes: collaborative agents, interface agents, mobile agents, information agents (aka Internet agents), reactive agents, and hybrid agents [87][133].

Heucke writes that an "agent should show social behavior and should feature the ability to interact with external entities" [87]. When a group of intelligent agents collaborate to solve a problem, we call this a multiple agent system (MAS). Jennings defines this as "a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [96, 17].

The concept of employing multiple agent systems is not new, and scholars have proposed various ways to characterize and classify these systems. One way to look at a MAS is to examine the nature of the interactions among agents. Jennings et al. name three common types of interactions among agents: cooperation, coordination, and negotiation [96, 9].

A big challenge of multiple agent systems is "to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate their actions" [96, 18].

However, the autonomous nature of individual agents with often differing goals and heuristics does not necessarily mean that, working together, they cannot collectively produce desirable outcomes. Jennings et al. write that "self-interested agents, by definition, simply choose a course of action which maximizes their own utility. In a society of self-interested agents, it is desirable that if each agent maximizes its local utility, then the whole society exhibits desirable behavior" [96, 24].

Polgar and Polgar agree: "The ultimate goal of cooperation and coordination is to reach a globally optimal solution independent of the language or protocol used" [139, 812]. They write "if we map the cooperation goals into distributed problem-solving strategies and let each agent play the role of a cooperating computational unit instead of an autonomous negotiator, it is then possible to deploy distributed constraint satisfaction problem-solving strategies" [139].

There has been limited published work on the marriage of multi-agent systems and evolutionary algorithms, but some have argued that there is much compatibility between multiple agent systems and Genetic Algorithms (GAs) that can be exploited. Sarker, when considering whether agent-based GAs are an emerging paradigm, notes first that "hybridization with another method or methods is a very common practice" when constructing GAs, and that the information sharing inherent in a MAS can be leveraged here as well. [156, 45]

"In a multiple agent system, the relevant information from individual agents are compiled, theories about their behavior are formulated ... resulting in an emergence of system properties and behaviors," Sarker writes. "These generic steps are not much different from the steps entailed by Evolutionary Algorithms. In other words, they are similar to any other rational problem-solving approaches" [156, 45].

He goes further to point out that multiple agent systems are useful when an explicitly defined fitness function is difficult to construct. It is this possibility of incorporating a multiple agent system into an Evolutionary Algorithm's heuristic function that we wish to explore. It is intriguing to ask if a group of agents, each of which is able to inform on part of the fitness of a solution, but none of which can inform on all of it, can collectively provide an admissible heuristic for the algorithm.

5.3.1 Pools of Cooperating Agents

We investigate creating one or more pools of agents, each concerned with a specific quality attribute, that together can inform on the overall quality of a solution. In this system, a single agent of a specific type is assigned to each district, and that agent assesses and renders a subjective quality assessment of that district only, without examining other districts in the solution. For example, a set of agents tailored to assess a metric, such as the equality of population distribution or contiguity of a district, can separately evaluate the suitability of the district to which they are assigned and, when combined together, provide an overall assessment of the entire candidate solution.

For agents that assess their district in terms of population equality, compactness, or a similar measure, the agent's heuristic is assessed along a Likert scale [118][124]. A Likert scale was selected in deference to the concept that each agent is a virtual stakeholder with a specific perspective and agenda for its district. If the agent is asked to provide an assessment that is analogous to the subjective assessment that might be produced by a human stakeholder — which would conceivably, or even likely, be measured on a Likert scale — it makes sense to apply the same for the agent's assessment.

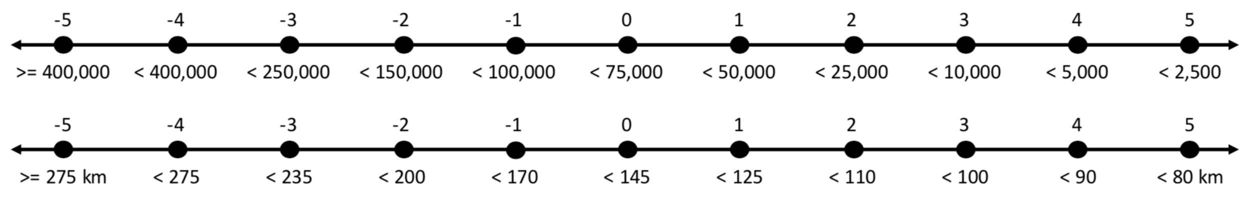

Figure 5.5: A sample Likert scale heuristic that might be used by agents for quality assessment of candidate solutions. Population equivalence can be assessed based on the difference between zone population size and ideal population size, top, and compactness can be assessed based on the average zone distance from the district center, bottom. The width of the scale or the granularity of divisions in the scale can be adjusted to provide more or less selective pressure, as desired. The position of the origin is arbitrary and does not affect performance.

Although it would be possible for agents to provide a continuous spectrum of values across all whole or real numbers, it is not necessary. Recall that the purpose of a fitness function is merely to order one candidate solution relative to another. The degree of inequality between two solutions is irrelevant; it is necessary only that they can be ranked relative to each other. "Finer granularity is not required," writes Fogel, and "criterion need not be specific with the precision that is required of some other methods" [63, 2].

The width of the Likert scale can be tailored to increase or decrease selective pressure and the scale need not be centered on the origin of the number line *(see Figure 5.5)*. The wider the scale, the less likely that two candidate solutions will receive an equivalent score from all agents, but in the event that an equality does occur, a secondary heuristic may be used as a tiebreaker.

For agents with a binary decision, such as district contiguity agents, the agent's heuristic returns a value of ± 1, although this value may be scaled when combined with the results from other agents with a greater magnitude of values.

Is it valid to call this technique an "intelligent agent?" This system meets Russell and Norvig's definition of "situatedness" for an agent, in that each agent is able to perceive the zones that comprise the district to which it is assigned and, by providing a quality assessment of that part of the solution, can effect changes on the environment through the collective ratings of all agents, which in turn affect the chances of the solution being selected for recombination or mutation and, therefore, including most of its genome in subsequent generations [154]. Additionally, this behavior is also exemplar of both Franklin and Graesser's definition of an agent pursuing an agenda that is specific to itself [72] and also Heucke and Gilbert et al.'s definition of an agent as a proxy for a user and his goals [87][78].

5.4 Results of Heuristic Evaluation by Intelligent Agents

Three trials were performed for an Evolutionary Algorithm with pools of intelligent agents assessing one or more quality objectives. The first trial employed a pool of seven agents — one assigned to each district — to assess the quality of population distribution within that district and a second pool to assess the contiguity of the zones in that district.

The population agents' heuristic measures the district population against an optimal population size (e.g. the total population divided by the number of desired districts) and then assesses the district based on difference between the district's population and the ideal: $f_{pop}(i) = \left| P_i - \frac{P}{k} \right|$, where P is the total population, k is the number of districts, and P_i is the population of district i. The population difference is then subjectively scored based on the Likert scale described above. Together, the pool of agents collaboratively provide a quality measure for the entire solution: $\sum_{1}^{k} L(i)$, where $L(i)$ is the Likert scale score of the agent for district i.

For the pool of contiguity agents, the heuristic is simpler: $f_{cont}(i) = -1$ or $+1$. And the collaborative assessment of all these agents becomes $\sum_{1}^{k} f_{cont}(i)$. For district compactness agents, a similar Likert scale and formula as that employed for population agents was utilized.

The agents are successfully able to rank order the members of the population and, thus, provide enough selection pressure through tournament selection for fit solutions to reproduce more frequently and, over time, increase the overall fitness of the population *(see Table A.21)*. One sample solution produced depicts a partitioning of the state into seven

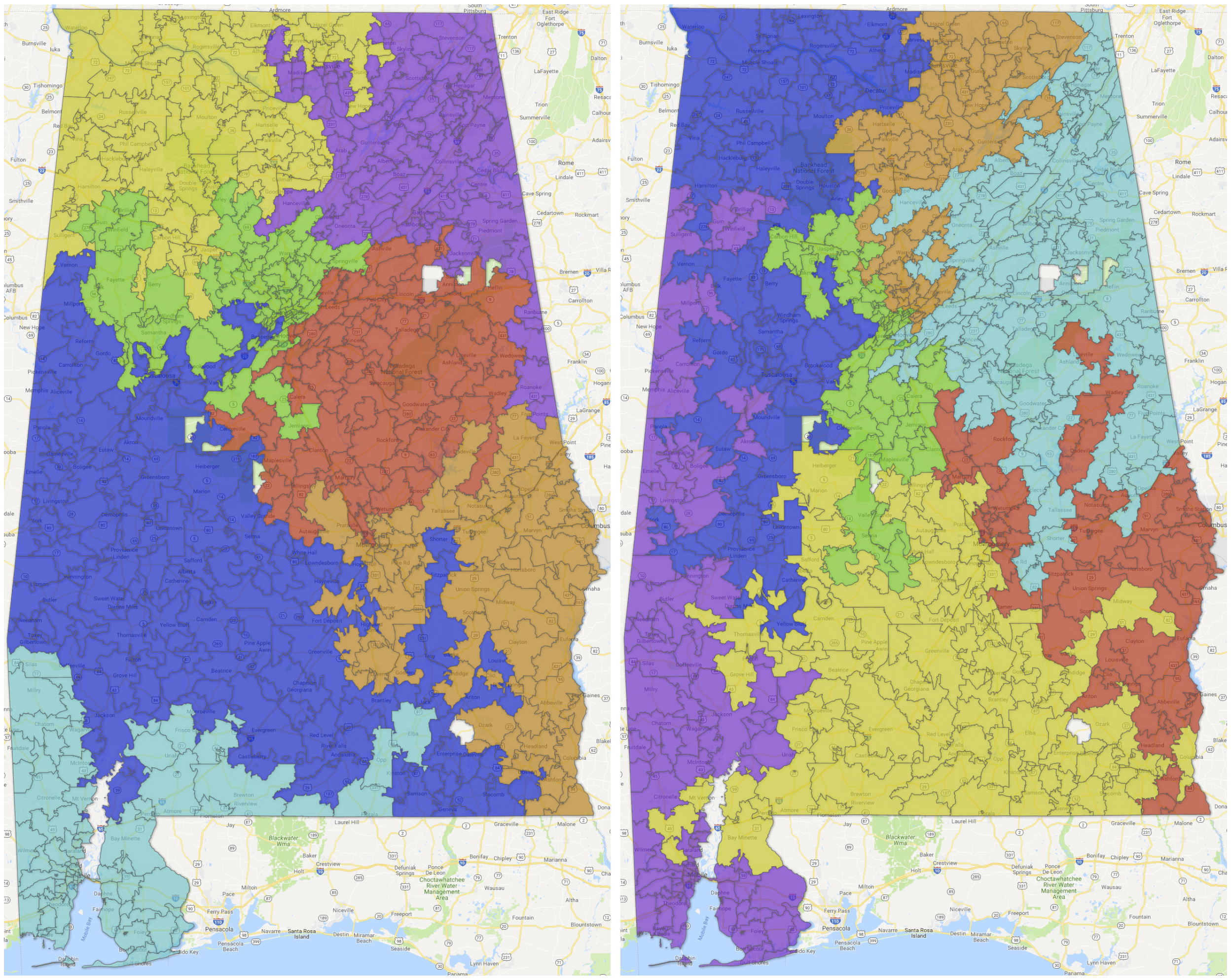

Figure 5.6: Two near-optimal solutions for districting the state of Alabama into seven contiguous zones using pools of agents to provide a quality heuristic. The first map, left, depicts an equal distribution of total population with a difference of no more than 136 persons between any two districts. The second map, right, depicts equal distribution of non-white population with a difference of no more than 13 persons between any two districts. Both solutions were obtained using a pool of seven agents, one dedicated to each district, working together to provide an overall quality heuristic for the entire solution. The algorithm employed 100 iterations of a mutation only evolutionary algorithm running for 500 generations with a mutation rate $p_m = 0.005$, a population size $\mu = 100$, randomly generated district centers, and Neighborhood Mutation with Local Repair.

contiguous districts with a maximum population difference between any two zones of 36 persons *(see Figure 5.6)*.

A second trial repeats this process with the substitution of a pool of agents that assess the distribution of non-white, or minority, individuals within the district compared to the ideal distribution: $f_{minority}(i) = \left|P_i - \frac{P}{k}\right|$, where P_i is the non-white population of the district. These agents perform similarly to the first trial *(see Table A.21)*, and produced an extraordinarily fit solution with a maximum non-white population difference between any two districts of only 13 persons *(see Figure 5.6)*.

Finally, a multi-objective trial was conducted with four separate pools of agents. Each pool of seven agents, with one agent assigned per district, assesses each of the four quality criteria: equality of total population distribution, equality of minority population distribution, district compactness, and district contiguity. Together, a total of 28 agents independently assess districts according to their own criteria and combine these assessments to produce a single quality measure for the entire solution. The individual measures produced by each pool were summed, with a scaling factor of 5.0 applied to the contiguity agents' score to provide a difference in maximum and minimum scores equivalent to that of the other pools.

This trial produces results comparable to that seen in the weighted multi-objective heuristic seen previously, with similar improvement in mean population fitness and standard deviation of population fitness over successive generations *(see Table A.22)*. One candidate solution achieved a total population equality of no more than 370 persons with a minority population inequality of no more than 44,991 persons and a maximum average distance from district center of 337.0 km *(see Figure 5.7)*.

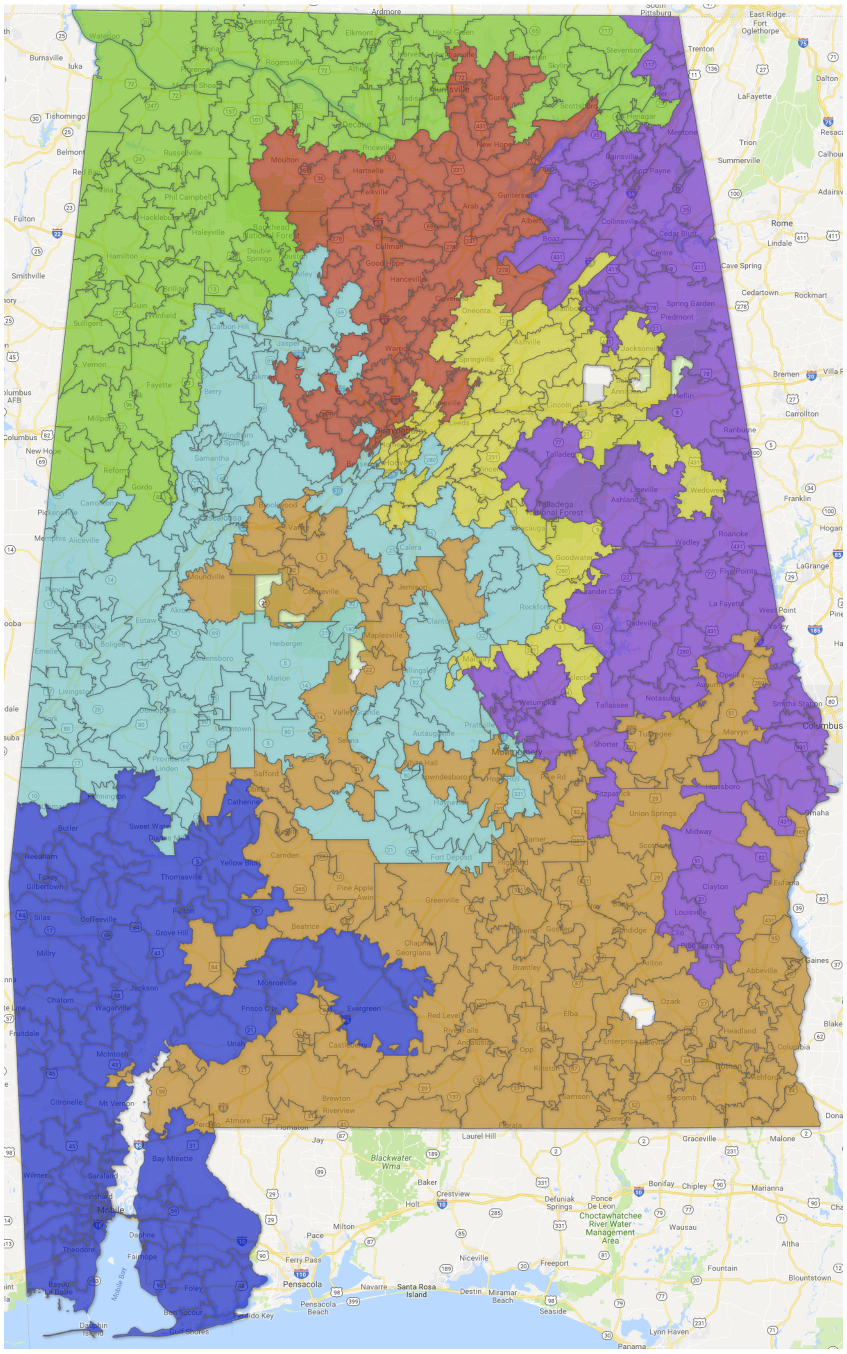

Figure 5.7: A candidate solution for districting the state of Alabama into seven contiguous zones using four collaborative pools of heuristic agents to separately assess total population distribution, non-white population distribution, district compactness, and district contiguity. The four separate pools of agents' heuristics are then combined to rank order the population. This map depicts a potential solution with a total population difference of no more than 370 persons between any two districts, a non-white population difference of no more than 44991 persons between any two districts, and an average distance from district center of no more than 337.0 km. The algorithm employed 100 iterations of a mutation only evolutionary algorithm running for 500 generations with a mutation rate $p_m = 0.005$, a population size $\mu = 100$, randomly generated district centers, and Neighborhood Mutation with Local Repair.

Chapter 6
Conclusions and Future Work

"*Writing is like driving at night in the fog. You can only see as far as your headlights, but you can make the whole trip that way.*"

— E.L. Doctorow [138]

6.1 Conclusions on Evolutionary Solutions to Districting Problems with Contiguity Constraints

Solving basic districting problems with evolutionary algorithms is no more or less challenging than most other evolutionary optimized search problems. However, there are two additional complications for evolutionary districting problems that can make evolutionary searches difficult: incorporating multiple, independent heuristics; and obeying a geographic contiguity constraint. The latter is the more challenging, and has been dealt with in multiple ways, including simply discarding non-contiguous candidate solutions [29][86] or by imposing a fitness penalty [12][44].

The destructive effects of crossover on the genetic string for contiguous districting problems are severe. Random crossover produces an infeasible solution in almost every case; multiple- and single-point crossover are less destructive, but still produce infeasible solutions a majority of the time. The effects of individual mutations on the genetic string are potentially as severe, but the less frequent occurrence of mutation, controlled by a mutation factor that is typically set at a value of once per every hundred or thousand genes, makes this more manageable.

With adequate selective pressure, a simple Genetic Algorithm can, in many cases, overcome both of these negative tendencies and still produce a reasonably fit solution, as observed in the performance of a typical, unoptimized genetic algorithm on the small-scale districting problem for Jefferson County, Alabama *(see Figure 4.2 and Table 4.3)*. However, the percentage of the population that are infeasible solutions is high and these impede the rate of improvement of the population as it converges toward a highly fit solution.

We conclude that a simple genetic algorithm, with traditional $k = 2$ tournament selection, single-point crossover, and stochastic mutation with rate $p_m = 0.001$ can produce a reasonably fit solution to a basic districting problem with a geographic contiguity constraint. Although algorithm performance is not as efficient as seen in later, more sophisticated algorithms, the observed results seem sufficient to at least partially validate Hypothesis 1.

As seen in trials of the algorithm operating in a 25x25 grid "sandbox" with simulated demographic data, the size of the population and the rate of mutation both have a strong effect on algorithm performance. We see that population sizes between 100 and 500 individuals allow sufficient genetic variety in the population for a mutation-based algorithm to converge to highly fit solutions despite more than 53% of the population being comprised of non-contiguous, invalid, candidate solutions. We also observe that the percentage of discontiguous solutions varies inversely with the population size — that is, as the population size decreases, the percentage of invalid solutions in the population increases, with a corresponding negative impact on the quality of the solution found *(see Table A.3)*.

The implementation of the Neighborhood Mutation algorithm in place of purely stochastic mutation results in a vast increase in the percentage of valid solutions in the population for all population sizes and in all generations for the duration of algorithm execution —

more than 99% in all cases. The quality of solutions found with Neighborhood Mutation is correspondingly higher, and the rate of convergence is likewise faster *(see Table A.4)*.

In lieu of Neighborhood Mutation, traditional, stochastic mutation can be paired with the Local Repair algorithm, which attempts to detect and correct contiguity errors after they occur. Observations show that Local Repair is even more effective then Neighborhood Mutation at the traditionally accepted mutation rate of $p_m = 0.001$ [81][126], resulting in a population with more than 99.8% valid solutions at each population size and for all generations throughout the algorithm's execution *(see Tables A.5, A.8, A.9, and A.11)*. Solution quality and rate of convergence are both comparable to that found with Neighborhood Mutation.

When mutation rates are increased above the traditional value, solution quality and algorithm convergence can increase up until a point at which destructive mutations occur in the genome at a rate too high for the Neighborhood Mutation algorithm to prevent and for the Local Repair algorithm to correct. Observations suggest that, for Neighborhood Mutation alone, this rate is approximately $p_m = 0.010$ for the type of algorithm and the districting problem investigated here *(see Table A.7)*. When Neighborhood Mutation is combined with Local Repair, the effects of higher mutation rates can be significantly lessened but not eliminated *(see Table A.9)*.

We conclude that, in terms of solving a districting problem with geographic contiguity constraint on a state-wide level, both Hypothesis 2 and Hypothesis 3 are validated by the observed data. This similarly strengthens validation of Hypothesis 1, which we consider confirmed.

6.2 Conclusions on Multi-Objective Heuristic Evaluation of Evolutionary Districting Problems

We show that an evolutionary algorithm can perform well with a weighted multi-objective heuristic and produce multiple, reasonable candidate solutions to a districting problem while balancing two or three separate objectives that are relatively non-conflicting. Observations demonstrate that a weighted, multi-objective heuristic can provide adequate selective pressure to allow a population to continually improve and converge.

This holds true for a weighted heuristic that includes both total population distribution with district compactness and contiguity *(see Tables A.13 and A.14)* and minority population distribution with district compactness and contiguity *(see Tables A.15 and A.16).* In both instances, mean population fitness shows a continued downward trend over time for various mutation rates and each also shows a consistent downward trend in the standard deviation of the population fitness over time, demonstrating convergence of the algorithm.

However, with the introduction of a second population distribution measure, the ability of the weighted multi-objective heuristic function to simultaneously provide adequate selective pressure to each objective becomes questionable. Trials involving both a total population and minority population distribution objective, with a soft district compactness constraint, show an ability to consistently but slowly improve the population fitness over time, but the standard deviation of population fitness fails to tighten *(see Tables A.17 and A.18).* The addition of a fourth objective, such as district compactness, produces similar results *(see Tables A.19 and A.20).*

Replacement of the weighted multi-objective function with a pool of heuristic agents is especially effective, with a very quick improvement in population fitness for trials involving a total population or minority population objective and a contiguity constraint *(see Table A.21).* In both trials, both the mean population fitness and standard deviation of population fitness have pronounced downward slopes, demonstrating strong selective pressure and convergence. Both trials produce reasonable candidate solutions comparable to those found in non-agent approaches. A third trial employing four pools of agents — each focusing on either total population distribution, minority population distribution, district compactness, or district contiguity — produces results comparable to that of the weighted four-objective heuristic above *(see Table A.22).* Compared to the weighted multi-objective heuristic, the multiple pools of agents show similar improvement in mean population fitness over time. However, the trial with multiple pools of agents shows a more pronounced downward slope to the standard deviation of fitness over time, suggesting, but not conclusively, that the multiple agent pool approach may better promote algorithm convergence in this instance.

It is worth noting, however, that the selective pressure is dependent on a secondary heuristic that can be used to resolve ranking ties (e.g. when the collective agents' score of two candidate solutions is equal). The frequency of ties is dependent on the width of the Likert scale used by each specific type of agent, but ties do occur relatively often, but with a greater frequency in early and later generations. Because a randomly generated initial population is usually of a uniformly poor fitness, ties tend to occur more often in early generations of the algorithm. Similarly, as the population converges and becomes less genetically diverse, ties are more apt to occur in later generations as well. There are multiple feasible approaches to resolving such ties: a coin toss, prioritizing one agent pool's heuristic over another, or employing a secondary objective function. Each will work, but the choice of method affects both the selective pressure and convergence rate of the algorithm.

Another observation is that the contiguity agents' heuristic, which is a binary ± 1 can be overwhelmed by the magnitude of the rating scale for other agent pools, and therefore a weighting factor for the contiguity agent pool may be desirable to prevent dominance by other pools and a subsequent increase in infeasible solutions in the population.

We observe that employing a pool of multiple, independent agents that inform separately on discrete parts of a candidate solution can then merge those assessments to produce a combined heuristic rating for the entire solution. This is demonstrated in trials involving pools of different types of population agents *(see Figure 5.6)*, which we consider to confirm Hypothesis 4-A. The ability of multiple pools of agents with different quality heuristics to work collaboratively to achieve a goal is similarly demonstrated in Table A.22 and Figure 5.7, which we consider to confirm Hypothesis 4-C for this type of problem.

There is anecdotal evidence to suggest that solutions that are found to be partially infeasible by one or more agents can evolve into feasible solutions in subsequent generations. The observation that the percentage of feasible solutions in agent-based trials — particularly those involving many competing agent pools — can decrease and then later increase suggests that some number of infeasible solutions can mutate and become feasible, but the effects of stochastic selection and mutation likely account for much of this variation. Therefore, we do not feel that adequate data is available to confirm or deny Hypothesis 4-B, which we aim to resolve through more granular future studies.

6.3 Future Work

Through the development of this algorithm, a number of difficulties and opportunities for research arose that are worthy of consideration in future studies:

Reducing the Destructiveness of Crossover The effect of single-point, two-point, and especially uniform crossover on the genome for a contiguous districting problem is disastrous. In the vast majority of cases, such crossover is fatally disruptive in terms of district contiguity. It would be worthwhile to investigate alternative approaches to crossover that are less likely to render an otherwise fit solution infeasible. One particular approach that may bear fruit is the implementation of a non-linear genome with position-independent genes, much like that described by Holland's original inversion operator [92], that can be reordered in the genome without losing the mapping of a specific gene to a specific district. This would allow the clustering of genes allocated to a specific district close to each other and, thus, allow crossover to occur at a point within a district, and thus not disrupt it.

Growth of Infeasible Solutions The rapid increase in the percentage of infeasible (i.e. discontiguous) solutions produced with high mutation rates ($p_m \geq 0.01$) could perhaps be mitigated through a combination of several methods: the widening of the neighborhood examined by the neighborhood mutation algorithm to reduce the number of incompatible adjacent mutations, the implementation of the local repair algorithm with a similarly widened area of examination for discontinuity detection, and the employment of elitism with either a generational or steady-state evolutionary algorithm to ensure that more-fit solutions are not crowded out of the population.

Weighting for Multi-Objective Heuristic Finding suitable weights in a multi-objective heuristic function that will produce a set of solutions along a the Pareto frontier, along which no single objective tends to dominate over another, can be a significant problem for Genetic Algorithms in general [20]. It is, in particular, a significant challenge for the type of districting problem explored here. There is research to suggest that a genetic algorithm can simultaneously coevolve a set or sets of weights for a multi-objective heuristic that can produce both non-dominated solutions that are also subjectively acceptable to the user [20]. Other approaches to determining a desirable values for a weighted multi-objective function are well researched and could provide clarity on both the weighted mutli-objective heuristic trials and the multi-objective agent-based heuristic trials attempted here [31][43][97][186].

Modeling User Preferences in Agent Heuristic Finally, the potential of intelligent agents to act as a proxy or surrogate for a human stakeholder can be further investigated through attempts to map agent performance to human-subject preferences with respect to various objectives or subjective quality criteria for districting problems. This could lead to the development of more representative agent models that can function as a proxy in a manner similar to that envisioned by Chou et al.'s VSF.

Glossary

admissible heuristic A heuristic that never overestimates the the actual cost of reaching a goal. Also sometimes called an “optimistic heuristic”. 143, *see also* heuristic function

agent An agent is anything that can be viewed as perceiving its environment, through sensors or other means of input, and acting upon that environment through actuators or other means of output [116]. ii, iii, 5, 6, 8, 139, 142–145, 147, 161, *see also* rational agent

agent function In artificial intelligence, a mathematical or other function that take a series of perceptions of the external environment and maps them to an action or sequence of actions to be performed by the agent as a response. An agent function may or may not incorporate past perceptions in its calculations.

allele A shortened form of the word **allelomorph**, one of two or more possible variant forms or values of a gene at a particular locus along a chromosome [81, 28]. Allelic variation at a specific locus is defined as the number of distinct alleles present in a population [148]. 11, 12, 15, 19, 25, 27, 32, 38, 59, 62, 63, 78, 82, 88, 89, 94, 96, 104–108, 156–159, 161, *see also* gene

allelomorph An **allele**. 15, 156, *see* allele

annealing In computer science, a heuristic function that is used to approximate global optimization in a search space that, in simplistic terms, evaluates “nearby” solutions and moves toward solutions that are closer to the theoretical optimum. Also referred to as *simulated annealing*. 22, 68, 76

certainty factor In artificial intelligence, a numerical value, usually in the range 0..1 that represents the level of confidence that an assertion is true.

chromosome A molecule that contains all or part of the genetic information of an organism [100]. 13, 15, 19, 55–60, 63, 77, 78, 101, 102, 104, 156–159

convergence In evolutionary programming and genetic algorithms, the phenomenon in which every individual in the population is identical [114]. 13, 48, 53, 60, 61, 64, 65, 68, 70, 97–99, 103, 105, 107, 109, 134, 151–153

deterministic A system in which randomness is not involved, or for which a sequence of actions or calculations will always be consistent for the same starting condition. Deterministic systems are often considered to be stateless, since past conditions have no effect on future actions. 23, *see also* stochastic

diploid In genetic algorithms, a diploid genetic algorithm is one in which each member of the population has more than one chromosome in which alleles of genes are also assigned additional characteristics, or values, such as "dominant" or "recessive," that affect how those genes are involved in the mating process. More simply, diploid means that genetic chromosomes are paired. Diploid GAs therefore differ from traditional algorithms in both its representation of the genome and also its reproduction operations [135] [114, 29]. 12, 16, 31, *see also* haploid

disruption In evolutionary programming or genetic algorithms, a schema is said to be disrupted if a child of an parent that matches a schema H does not itself match H [114]. 27, 28, 55, 57, 59, 63,

elitism In genetic algorithms, particularly steady-state GAs, the practice of never discarding a more fit solution to make room in the population for a less-fit child solution [167, 25]. 16, 32, 40, 43, 51–53, 77, 80, *see also* infant mortality

epistasis Non-linear interaction between genes. The term epistasis describes a certain relationship between genes, where an allele of one gene hides or masks the visible output, or phenotype, of another gene. This should not be confused with the concept of dominant and recessive genes, which are terms that apply to different alleles of the same gene [160] [114]. 11, 15, *see also* gene

evaluation function A calculation of the cost or value for a given state or node in an informed search algorithm. For example, in a best-first search, the node with the lowest evaluation function result might be deemed the best node and, therefore, would be selected for expansion. , *see also* heuristic function

frontier The current set of all leaf nodes in a graph or tree. Search-based problem solving algorithms generally expand the set of known states by performing actions on leaf nodes to create additional known states [153, 75].

gamete A single chromosome from one haploid parent that fuses with another haploid during recombination to produce a zygote [126, 5]. , *see also* chromosome

gene A unit of hereditary information at a specific location, called the locus, along a chromosome, which is composed of many separate genes. In a genetic algorithm, a gene represents a specific value at a specific location in the chromosomal sequence of an individual in the population [60]. 15–17, 19, 25–27, 30, 31, 36–38, 54–63, 77, 78, 88, 89, 96, 101, 102, 105, 156, 158–160, *see also* locus

Genetic Programming An extension of the concepts of evolutionary computation and genetic algorithms into the realm of computer program construction. In this approach, computer programs representing an executable solution to a problem, are generally expressed as trees of discrete operations and operands. These trees are modified and added onto through methods similar to those found in genetic algorithms to create new, potential more accurate, programs to solve the problem [37]. 11

genome The complete set of genes or genetic material present in a cell or organism. In evolutionary computing, a genome is typically represented as a string of binary or higher order values that, together, encode a representation of a potential solution for the problem to be solved. 19, 25–27, 32, 36, 37, 39, 42, 53, 55, 58, 60, 77, 78, 87, 96, 145, 151, 154, 157, 159, *see also* gene

genotype A genotype is the entire genetic makeup of an individual, or the complete list of genes present in an individual. But the term is sometimes more narrowly construed to mean the set of alleles present at one or more loci in a chromosome. In genetic algorithms, genotype generally refers to the string or array of values that represent a potential solution to a problem [25] [114]. 11, 60, 159, 160, *see also* phenotype

Hamming distance In Genetic Algorithm with binary chromosomes, the Hamming distance between two genomes can be calculated as the number of genes that differ between the two genomes. Two individuals who are identical with the exception of a single gene would have a Hamming distance of 1 between their genomes. Individuals with two differing genes would have a Hamming distance of 2, and so on [83] [126]. 19, 58

haploid In biology, a cell that has only one chromosome set. In genetic algorithms, the term haploid implies that each potential solution to a problem has only a single string or array of values representing the full genotype of the individual [4] [114]. 31, 158, *see also* diploid

heuristic Generally speaking, a "rule of thumb" measure when making decisions or classifications in an algorithm [46]. ii, iii, 4, 5, 8, 20, 122, 160, *see* heuristic function

heuristic function Such functions are a common means by which additional understanding of a problem domain can be incorporated into a search algorithm. 2, 143, 156, *see also* admissible heuristic

hyperspace A multi-dimensional space of greater than three dimensions. A related concept is that of a hypercube, which is a generalization of a three-dimensional object into more than three dimensions. A hypercube of n dimensions is called an "n-cube"[176]. 18, 25, 129

infant mortality In genetic algorithms, the practice of never allowing an invalid or far inferior child solution to enter the population. 53, 54, *see also* elitism

Keyhole Markup Language A markup language based on the Extensible Markup Language (XML) that is used to express single or nested geographic locations, boundaries, polygons, and other annotations. Originally developed by Keyhole, Inc., it has since been adopted as a voluntary international standard by the Open Geospatial Consortium. 85, 109, 164

locus In genetic algorithms, the locus refers to the position in a genetic string of a particular gene [114]. 31, 36, 38, 55–57, 59

metaheuristic In general, a problem-independent framework, or general algorithm, that is used to develop a tailored, more specific algorithm or solution to an individual problem. The term is sometimes loosely used to refer to a specific implementation of a heuristic optimization algorithm [162]. 130, *see also* heuristic

orgy In genetic algorithms, a method for creating child instances by recombination of the chromosomes of all individuals in the population at the same time, as opposed to the recombination of the chromosomes of two parents only [167, 24]. 38

phenotype The visible characteristics or traits of an individual. In the field of genetics, an organism's genotype is the genetic information that manifests itself in the form of its phenotype. 11, 15, *see also* genotype

propagation In genetic algorithms, propagation refers to the inheritance of the characteristics of one generation into subsequent generations. More broadly, a schema is propagated if individuals in the current population match that schema as well as individual in the subsequent population. A schema can propagate even if the individuals that match it are not children of the parents that match it, although they often are [114]. 1, 6, 29, 37, 39, 42, 43, 80

pruning The concept of removing from consideration in a search algorithm one or more possible solutions, particularly in tree- or graph-based problems, without having to first examine them. 52

rational agent An agent that acts in a manner that achieves, or aims to achieve, the best possible outcome, or when there is uncertainty, the best expected outcome [153, 4]. , *see also* agent

satisficing The goal of finding a course of action that is 'good enough,' as opposed to the sometimes unrealistic goal of finding the absolute best or 'maximum' solution. i.e. "replacing the goal of *maximizing* with the goal of *satisficing*" [161].

schema Also sometimes called a similarity template, a definition of a set of linear values that members of the conforming set must match. Schemata may be of varying lengths, and some positions in the defining template may be "unknown" or "don't care". Therefore, given m possible values, or alleles, at a specific position in a string, there are $m + 1$ possible values for that position in the schema. A schema is characterized both by its *order* and its *defining length*, in addition to the specific values at each position in the schema. Given a schema with a defining length of n, there are n^m possible permutations of that schema [81, 29] [114]. 26–30, 39, 55–58, 92, 157, 161

selective pressure In a genetic algorithm, the probability of the best individual being selected compared to the average probability of selection of all individuals. Increasing selective pressure results in a faster loss of population diversity [16][178]. 47–50, 53, 115, 132, 134, 144, 150, 152, 153

stochastic A random process or function or one employing randomness. Examples of stochastic processes include random walks and Bernoulli, Poisson, Markov, and Gaussian processes. Most genetic algorithms are stochastic in design [135]. 4, 12, 15, 21, 29, 30, 36, 43, 44, 47, 60, 62, 90, 94, 109, 111, 150, 151, *see also* deterministic

syllogism A deductive scheme of a formal argument consisting of a major and a minor premise and a conclusion (as in "every virtue is laudable; kindness is a virtue; therefore kindness is laudable")[1].

takeover In a genetic algorithm, the phenomenon of an entire population being comprised of individuals identical or substantially similar to a single individual which has has existed in some form since the initial population [40]. 54, 162

takeover time The amount of time, usually measured in generations, for takeover to occur in a genetic algorithm [40]. , *see* takeover

tractability The ability to solve or make progress in solving a problem in a reasonable time. In reference to computational problem solving, a problem or type of problem for which the time required to solve a case grows exponentially with the size of the problem instance is referred to as *intractable* [153, 8].

ZIP Code Tabulation Area A geographic block designated by the United State Census Bureau that is comprised of addresses from one or more ZIP codes. ZCTAs are based on census block boundaries, and not those of the U.S. Postal Service ZIP code area. In many cases ZCTAs and Zip codes are very similar, and as a general rule the majority of addresses within a ZCTA share the same ZIP code. However, some addresses from other zip codes may be included in a ZCTA with a different identifier based on census block boundaries. Demographic data from the decennial census is aggregated by ZCTA, and is not available based upon ZIP code alone [171][94]. 85, 109, 114, 126, 165

Acronyms

BGA breeder genetic algorithm. 45,

BTS Binary Tournament Selection. 42–44, 51, 52

EA Evolutionary Algorithm. ii, iii, xv, 2, 5, 7, 17, 24, 25, 34, 64, 65, 81, 84, 97, 100, 130, 133, 143, 145

EC Evolutionary Computation. 1–3, 10, 11, 13, 16, 17, 19, 25, 34, 52–55

EP Evolutionary Programming. 10, 12, 16,

ES Evolutionary Strategies. 10, 12, 13, 16, 35, 54,

EVOP Evolutionary Operation. 11,

FPS Fitness Proportionate Selection. 42, 43, 45, 47

GA Genetic Algorithm. 1, 2, 5, 8, 10–13, 16, 24–26, 30–35, 41, 53, 54, 68, 76–80, 94–96, 101, 130, 142, 150, 155, 157, 159

HLAI human level artificial intelligence.

HUX half-uniform crossover. 58,

IEC Interactive Evolutionary Computation. 83

KML Keyhole Markup Language. 85, 109

MAS multiple agent system. 5, 141–143

RTS Roulette Tournament Selection. 43, 52,

RWS Roulette Wheel Selection. 43, 45–47, 51, 52, 77

SPX simplex crossover.

TS Tabu Search. 23, 68, 72, 75, 76, 78,

VSF validated surrogate fitness. 83, 155

ZCTA ZIP Code Tabulation Area. 85, 95, 109, 110, 112, 114, 126, 186

Bibliography

[1] Syllogism. In *Merriam Webster's Collegiate Dictionary*. Merriam-Webster, Inc., Springfield, MA, eleventh edition, September 2017.

[2] E.H.L. Aarts and P.J.M. van Laarhoven. Simulated annealing: An introduction. *Statistica Neerlandica*, 43(1):31–52, 1989.

[3] Michael Affenzeller. A generic evolutionary computation approach based upon genetic algorithms and evolution strategies. 28:59–72, 04 2002.

[4] Michael Allaby, editor. *Haploid*. Oxford University Press, third edition, 2006.

[5] Lee Altenberg. Advances in genetic programming. chapter The Evolution of Evolvability in Genetic Programming, pages 47–74. MIT Press, Cambridge, MA, USA, 1994.

[6] Lee Altenberg. The schema theorem and price's theorem. *Foundations of Genetic Algorithms*, 3:23–49, 1995.

[7] Micah Altman. Is automation the answer: The computational complexity of automated redistricting. *Rutgers Computer and Law Technology Journal*, 23:81–142, 1997.

[8] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[9] Joe Wirt Atmar, III. *Speculation on the Evolution of Intelligence and Its Possible Realization in Machine Form*. PhD thesis, Las Cruces, NM, USA, 1976. AAI7622497.

[10] Franz Aurenhammer. Voronoi diagrams&mdash;a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[11] Saad A. Azeem M. Modified queen bee evolution based genetic algorithm for tuning of scaling factors of fuzzy knowledge base controller. *Proceedings of the IEEE INDICON 2004*, pages 299–303, 2004.

[12] Fernando Bação, Victor Lobo, and Marco Painho. Applying genetic algorithms to zone design. *Soft Comput.*, 9(5):341–348, May 2005.

[13] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. In David B. Fogel, editor, *Evolutionary Computation: The Fossil Record*, pages 15–26. Wiley-IEEE Press, 1st edition, 1998.

[14] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, March 1993.

[15] James E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[16] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[17] Edwin Roger Banks, Paul Agarwal, Marshall McBride, and Claudette Owens. A comparison of selection, recombination, and mutation parameter importance over a set of fifteen optimization tasks. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 1971–1976, New York, NY, USA, 2009. ACM.

[18] William J. Baumol and Philip Wolfe. A warehouse-location problem. *Operations Research*, 6(2):252–263, 1958.

[19] David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms: Part 2, research topics. 15, 02 1970.

[20] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240, London, 1998. Springer London.

[21] Dr Prakash Bethapudi, Sreenivasa Reddy, T Sitamahalakshmi, and VARMA KAMADI. Feature analysis and classification of bi-rads breast cancer using genetic algorithm. 6, 03 2015.

[22] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, Mar 2002.

[23] Tobias Blickle and Lothar Thiele. *A Comparison of Selection Schemes Used in Genetic Algorithms*. Swiss Federal Institute of Technology, Zurich, Switzerland, December 1995.

[24] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computing*, 4(4):361–394, December 1996.

[25] PDQ Cancer Genetics Editorial Board, editor. *Genotype*. National Cancer Institute, 2016.

[26] G. E. P. Box. The exploration and exploitation of response surfaces; some general, considerations and examples. *Biometrics*, pages 16–60, 1954.

[27] George E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 6(2):81–101, 1957.

[28] George E. P. Box and Norman Richard. Draper. *Evolutionary operation; a statistical method for process improvement [by] George E. P. Box [and] Norman R. Draper.* Wiley New York, 1969.

[29] Burcin Bozkaya, Erhan Erkut, and Gilbert Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26, 2003.

[30] F. Z. Brill, D. E. Brown, and W. N. Martin. Fast generic selection of features for neural network classifiers. *IEEE Transactions on Neural Networks*, 3(2):324–328, Mar 1992.

[31] Jason Brownlee. *Clever Algorithms: Nature-Inspired Programming Recipes.* Lulu Press, Inc., Morrisville, NC, June 15 2012.

[32] Glen Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press, 2012.

[33] M. G. Bulmer. *The mathematical theory of quantitative genetics / M.G. Bulmer.* Clarendon Press ; New York : Oxford University Press Oxford, 1980.

[34] George H. Burgin. Systems identification by quasilinearization and by evolutionary programming. *Journal of Cybernetics*, 3(2):56–75, 1973.

[35] Carnegie Mellon University. AI FAQs: What is evolutionary programming (EP)?

[36] Carnegie Mellon University. AI FAQs: What's an evolutionary strategy (ES)?

[37] Carnegie Mellon University. AI FAQs: What's genetic programming (GP)?

[38] Daniel Joseph Cavicchio Jr. *Adaptive search using simulated evolution.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, August 1970.

[39] Center for Economic Studies, United States Census Bureau. Decennial census of population and housing.

[40] Uday Kumar Chakraborty, Kalyanmoy Deb, and Mandira Chakraborty. Analysis of selection algorithms: A markov chain approach. *Evol. Comput.*, 4(2):133–167, June 1996.

[41] P. K. Chawdhry and R. K. Pant, editors. *Soft Computing in Engineering Design and Manufacturing.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1997.

[42] J. Cheng, G. G. Yen, and G. Zhang. A many-objective evolutionary algorithm with enhanced mating and environmental selections. *IEEE Transactions on Evolutionary Computation*, 19(4):592–605, Aug 2015.

[43] Oliver Chikumbo. Using different approaches to approximate a pareto front for a multiobjective evolutionary algorithm: Optimal thinning regimes for eucalyptus fastigata. *International Journal of Forestry Research*, page 27, 2012.

[44] Christine Chou, Steven Kimbrough, John Sullivan-Fedock, C. Jason Woodard, and Frederic H. Murphy. Using interactive evolutionary computation (iec) with validated surrogate fitness functions for redistricting. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 1071–1078, New York, NY, USA, 2012. ACM.

[45] Jon Christensen, Joe Marks, and Stuart Shieber. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph.*, 14(3):203–232, July 1995.

[46] Per Christensson. Software terms: Heuristic.

[47] C.A.C. Coello. *A Survey of Constraint Handling Techniques Used with Evolutionary Algorithms.* Laboratorio Nacional de Informática Avanzada, 06 1999.

[48] Jared L Cohon. *Multiobjective programming and planning.* Mineola, N.Y. : Dover Publications, 2003. Originally published: New York : Academic Press, 1978. in series: Mathematics in science and engineering. With new pref.

[49] Leon Cooper. Location-allocation problems. *Operations Research*, 11(3):331–343, 1963.

[50] Elon S. Correa, Maria Teresinha A. Steiner, Alex A. Freitas, and Celso Carnieri. A genetic algorithm for the p-median problem. In Lee E. Spector and Erik D. Goodman, editors, *Proc. 2001 Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1268–1275, San Fracisco, USA, July 2001. Morgan Kaufmann.

[51] Michael Crichton. *Prey.* Harper Collins, New York, NY, USA, 2003.

[52] J.F. Crow and M. Kimura. Efficiency of truncation selection. *Proceedings of the National Academy of Sciences of the United States of America*, 76(1):396–399, January 1979.

[53] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* John Murray (publisher), London, England, 1859.

[54] Dilip Datta, Jose Rui Figueira, Carlos M. Fonseca, and Fernando Tavares-Pereira. Graph partitioning through a multi-objective evolutionary algorithm: A preliminary study. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 625–632, New York, NY, USA, 2008. ACM.

[55] Lawrence Davis and Martha Steenstrup. Genetic algorithms and simulated annealing: An overview. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 1, pages 1–11. Morgan Kaufmann Publishers Inc., Los Altos, California, 1987.

[56] R Dawkins. *The Blind Watchmaker.* Longman Scientific and Technical, 1986.

[57] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

[58] Kenneth Alan De Jong, David B. Fogel, and Hans-Paul Schwefel. A history of evolutionary computation. *Evolutionary Computation*, pages 40–59, 2000.

[59] René Descartes (1596-1650). *Discourse on method ; and, Meditations on first philosophy*. Hackett Pub. Co., Indianapolis, IA, third edition, 1993.

[60] Editors of Encyclopædia Britannica. Gene, heredity, September 22 2017.

[61] Larry J. Eshelman, Richard A. Caruana, and J. David Schaffer. Biases in the crossover landscape. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[62] D. B. Fogel and A. Ghozeil. Schema processing under proportional selection in the presence of random effects. *IEEE Transactions on Evolutionary Computation*, 1(4):290–293, Nov 1997.

[63] David B. Fogel. The advantages of evolutionary computation. In *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pages 1–11. World Scientific Press, 1997.

[64] David B. Fogel. *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1st edition, 1998.

[65] David B. Fogel. *An Introduction to Evolutionary Computation*. Wiley-IEEE Press, 1st edition, 1998.

[66] David B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. Wiley-IEEE Press, 2nd edition, 1999.

[67] David B. Fogel. In memoriam: Alex s. fraser (1923-2002). *IEEE Transactions on Evolutionary Computation*, 6(5), October 2002.

[68] David B. Fogel. Introduction to evolutionary computation. In Kwang Y. Lee and Mohammed A. El-Sharkawi, editors, *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*, pages 1–23. Wiley-IEEE Press, 2007.

[69] David Bruce Fogel. *Evolving Artificial Intelligence*. PhD thesis, La Jolla, CA, USA, 1992. UMI Order No. GAX93-03240.

[70] L.J. Fogel. *On the Organization of Intellect*. 1964.

[71] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK, 1966.

[72] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III: Agent Theories, Architectures, and Languages*, pages 21–35, London, UK, August 12-13 1996. Springer-Verlag.

[73] A. S. Fraser. Simulation of genetic systems by automatic digital computers. i. introduction. *Australian Journal of Biological Sciences*, 10:484–491, 1957.

[74] Juergen Gall, Bodo Rosenhahn, and Hans-Peter Seidel. *An Introduction to Interacting Simulated Annealing*, pages 319–345. Springer Netherlands, Dordrecht, 2008.

[75] R. S. Garfinkel and G. L. Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B495–B508, 1970.

[76] William V. Gehrlein. A comparative analysis of measures of social homogeneity. *Quality and Quantity*, 21(3):219–231, Sep 1987.

[77] Katrina Ellison Geltman. The simulated annealing algorithm, February 20 2014.

[78] Don Gilbert et al. The role of intelligent agents in the information infrastructure. *Proceedings of the 18th Annual Pacific Telecommunications Conference*, 1:912–919, January 14-18 1996.

[79] Fred Glover. Heuristics for integer programming using surrogate constraints. 8:156 – 166, 01 1977.

[80] Jeff Goddell. Inside the artificial intelligence revolution: A special report, pt. 1. *Rolling Stone*, February 29 2016.

[81] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[82] John J. Greffenstette and James E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 20–27, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[83] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[84] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. Evolution strategies. April 5 2013.

[85] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. *Evolution Strategies*, pages 871–898. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[86] S. W. Hess, J. B. Weaver, H. J. Siegfeldt, J. N. Whelan, and P. A. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965.

[87] Alexa Heucke, Georg Peters, and Roger Tagg. *Intelligent Software Agents*, pages 1598–1602. IGI Global, Hershey, PA, USA, 2005.

[88] Douglas R. Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid.* Basic Books, Inc., New York, NY, USA, 1979.

[89] John H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, July 1962.

[90] John H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[91] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[92] John Henry Holland. *Adaptation in Natural and Artificial Systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[93] Roy B. Hollstien. *Artificial Genetic Breeding Procedures for Parameter Optimization*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, September 1973.

[94] Phil Hurvitz. What is the difference between zip code "boundaries" and ZCTA areas?

[95] Christopher Ingraham. This is the best explanation of gerrymandering you will ever see, March 2015.

[96] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, Mar 1998.

[97] S. Jiang and S. Yang. An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts. *IEEE Transactions on Cybernetics*, 46(2):421–437, Feb 2016.

[98] Norman Johnson. Sewall wright and the development of shifting balance theory, 2008.

[99] D.F. Jones, S.K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1 – 9, 2002.

[100] Stephen Juan. What is the difference between a chromosome and a gene?, May 19 2006.

[101] Sung Jung. Queen-bee evolution for genetic algorithms. 39:575 – 576, 04 2003.

[102] Dervis Karaboga and Bahriye Akay. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.*, 31(1-4):61–85, June 2009.

[103] Ali Karci. Imitation of bee reproduction as a crossover operator in genetic algorithms, 01 2004.

[104] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[105] James Kennedy. *Particle Swarm Optimization*, pages 1113–1113. Springer US, Boston, MA, 2013.

[106] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[107] Abdullah Konak, David W. Coit, and Alice E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91(9):992 – 1007, 2006. Special Issue - Genetic Algorithms and Reliability.

[108] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[109] John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, Jun 1994.

[110] John R. Koza. Hidden order: How adaptation builds complexity. *Artificial Life*, 2(3):333–335, 1995.

[111] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

[112] Harold W. Kulin and Robert E. Kuenne. An efficient algorithm for the numerical solution of the generalized weber problem in spatial economics. *Journal of Regional Science*, 4(2):21–33, 1962.

[113] John E. Laird and Allen Newell. A universal weak method: Summary of results. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, pages 771–773, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[114] William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer Publishing Company, Incorporated, 1st edition, 2002.

[115] Jonathan W. Lartigue. An alternative approach to viral infection for solving the knapsack problem through evolutionary programming. Master's thesis, Auburn University, Auburn, Ala., April 2002.

[116] Svetlana Lazebnik. Lecture on rational agents. lecture, September 2017.

[117] Kwang Y. Lee and Mohammed A. El-Sharkawi. *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. Wiley-IEEE Press, 2007.

[118] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[119] Peter Lynch. How voronoi diagrams help us understand our world, January 2017.

[120] Robert E. Mahony and Robert C. Williamson. Prior knowledge and preferential structures in gradient descent learning algorithms. *J. Mach. Learn. Res.*, 1:311–355, September 2001.

[121] Luca Mariot and Alberto Leporati. Heuristic search by particle swarm optimization of boolean functions for cryptographic applications. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1425–1426, New York, NY, USA, 2015. ACM.

[122] Michael de la Maza and Bruce Tidor. An analysis of selection procedures with particular attention paid to proportional and boltzmann selection. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 124–131, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[123] John McCullock. Introduction to particle swarm optimization.

[124] Saul McLeod. The likert scale, 2008.

[125] K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer Academic Publishers, Boston, 1999.

[126] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1998.

[127] Noraini Mohd Razali and John Geraghty. Genetic algorithm performance with different selection strategies in solving tsp, 01 2011.

[128] Alcir J. Monticelli, Rubén Romero, and Eduardo Nobuhiro Asada. Fundamentals of simulated annealing. In Kwang Y. Lee and Mohammed A. El-Sharkawi, editors, *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*, pages 123–146. Wiley-IEEE Press, 2007.

[129] H. Mühlenbein, M. Schomisch, and J. Born. Paper: The parallel genetic algorithm as function optimizer. *Parallel Comput.*, 17(6-7):619–632, September 1991.

[130] Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evol. Comput.*, 1(1):25–49, March 1993.

[131] Heinz Mühlenbein and Hans-Michael Voigt. *Gene Pool Recombination in Genetic Algorithms*, pages 53–62. Springer US, Boston, MA, 1996.

[132] Giuseppe Narzisi. *Multi-Objective Optimization: A Quick Introduction.* Courant Insitute of Mathematical Sciences, New York University, New York, NY, USA, January 2008.

[133] Hyacinth S. Nwana. Software agents: an overview. *The Knowledge Engineering Review*, 11(3):205–244, 1996.

[134] A. K. Ojha and K. K. Biswal. Multi-objective geometric programming problem with weighted mean method. *International Journal of Computer Science and Information Security*, 7(2):82–86, 2010.

[135] A. Omidpour, B. Nasiri, K. Alagheband, and M. R. Meybodi. A new real-valued diploid genetic algorithm for optimization in dynamic environments. In *2014 Iranian Conference on Intelligent Systems (ICIS)*, pages 1–6, Feb 2014.

[136] G. Pavai and T. V. Geetha. A survey on crossover operators. *ACM Comput. Surv.*, 49(4):72:1–72:43, December 2016.

[137] Massimo Pigliucci. Sewall wright's adaptive landscapes: 1932 vs. 1988. 23:591–603, 01 2008.

[138] George Plimpton, Van Wyck Brooks, and Malcolm Cowley, editors. *Writers at Work: The Paris Review Interviews*. Penguin Books, second edition, November 1977.

[139] Jana Polgar and Tony Polgar. Designing agents with negotiation capabilities. *Encyclopedia of Information Science and Technology*, pages 810–815, 2005.

[140] R. Poli. Why the schema theorem is correct also in the presence of stochastic effects. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 487–492 vol.1, 2000.

[141] George R. Price. Selection and covariance. *Nature*, (227):520–521, 1970.

[142] George R. Price. Extension of covariance selection mathematics. *Annals of Human Genetics*, 35(4):485–490, 1972.

[143] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[144] Rosshairy Abd Rahman, Razamin Ramli, Zainoddin Jamari, and Ku Ruhana Ku-Mahamud. Evolutionary algorithm with roulette-tournament selection for solving aquaculture diet formulation. *Mathematical Problems in Engineering*, 2016, 2016.

[145] Ingo Rechenberg. *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Mit einem Nachwort von Manfred Eigen.* Frommann-Holzboog [Stuttgart-Bad Cannstatt], 1973.

[146] Charles S. ReVelle and Ralph W. Swain. Central facilities location. *Geographical Analysis*, 2(1):30–42, 1970.

[147] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, Jun 1973.

[148] Kara Rogers. What's the difference between a gene and an allele?

[149] Amos Ives Root, Ann Harman, Hachiro Shimanuki, and Kim Flottum. *The ABC and XYZ of Bee Culture: An Encyclopedia Pertaining to the Scientific and Practical Culture of Honey Bees.* A. I. Root Co., 41st edition, 2007.

[150] Jonathan E. Rowe. Genetic algorithm theory. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '07, pages 3585–3608, New York, NY, USA, 2007. ACM.

[151] Jonathan E. Rowe. Genetic algorithm theory. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 917–940, New York, NY, USA, 2012. ACM.

[152] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[153] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2nd edition, 2003.

[154] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, Upper Saddle River, NJ, USA, 3rd edition, 2010.

[155] Noureddin Sadawi. Java implementation of the roulette wheel selection method.

[156] Ruhul A. Sarker and Tapabraeta Ray. Agent-based evolutionary algorithms: Emerging paradigm or buzzwords? *OR/MS Today*, 38(5), October 2011.

[157] Y. Sawaragi, H. Nakayama, T. Tanino, A. Torokhti, and P. Howlett. *Theory of Multiobjective Optimization.* Mathematics in science and engineering. Academic Press, 1985.

[158] H.-P. Schwefel and J. Born. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. mit einer vergleichenden einführung in die hill-climbing- und zufallsstrategien. (isr 26) basel-stuttgart, birkhäuser verlag 1977. 390 s., sfr. 48,–. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 60(5):272–272, 1980.

[159] Hans-Paul Schwefel. Collective phenomena in evolutionary systems, 01 1987.

[160] Michael D. Shapiro. Pigeon breeding: Genetics at work.

[161] Herbert A. Simon. *Administrative behavior: a study of decision-making processes in administrative organization.* The Macmillan Company, 1st edition edition, 1947.

[162] Kenneth Sörensen and Fred Glover. Metaheuristics, 01 2013.

[163] William Spears. Crossover or mutation? 2, 07 1999.

[164] William Spears and Kenneth De Jong. On the virtues of parametrized uniform crossover. Technical report, Defense Technical Information Center, 01 1995.

[165] Erik Svensson and Ryan Calsbeek. *The Adaptive Landscape in Evolutionary Biology.* Oxford University Press, 2012.

[166] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, Sep 2001.

[167] Andrea Tettamanzi and Marco Tomassini. *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems.* Springer Publishing Company, Incorporated, 1st edition, 2001.

[168] The MathWorks Inc. Displaying a contour plot under a mesh plot.

[169] Tinh-Chi Tran, Tien Ba Dinh, and Viviane Gascon. Meta-heuristics to solve a districting problem of a public medical clinic. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, SoICT 2017, pages 127–134, New York, NY, USA, 2017. ACM.

[170] United States Census Bureau. Census tracts.

[171] United States Census Bureau. Zip code tabulations areas (ZCTAs).

[172] United States Census Bureau. ZIP code tabulation areas (ZCTAs). pamphlet, Washington, D.C., February 2015.

[173] United States National Archives. Enumeration district and related maps, 1880 - 1990.

[174] Abraham Wald. *Statistical Decision Functions.* Wiley: New York, 1950.

[175] Alfred Weber. *uber den Standort der Industrien, Translated as Alfred Weber's Theory of the Location of Industries.* PhD thesis, uber den Standort der Industrien, Tubingen, Germany, 1909.

[176] Eric W. Weisstein. Hypercube.

[177] Eric W. Weisstein. *Stirling Number of the Second Kind.* MathWorld. Last visited on 3/8/2018.

[178] Darrell Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. Morgan Kaufmann, 1989.

[179] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, Jun 1994.

[180] Ryan Wojes. What is annealing in metallurgy?

[181] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.

[182] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, June 1995.

[183] Sewall Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–366, 1932.

[184] Ningchuan Xiao. Geographic optimization using evolutionary algorithms. March 2003.

[185] Ningchuan Xiao, David A Bennett, and Marc P Armstrong. Using evolutionary algorithms to generate alternatives for multiobjective site-search problems. *Environment and Planning A: Economy and Space*, 34(4):639–656, 2002.

[186] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach, 1998.

Appendices

Appendix A

Additional Tables and Figures

24181	11233	19291	23499	4355	24974	18926	10903	6992	5344	12942	19372	21278	23719	10648	6939	10090	3147	10887	16934	24899	6809	4469	6432	12724
24789	5314	10038	11436	11486	6447	3181	19250	3434	17734	1129	12078	12604	14469	15811	9130	5170	16766	4839	22638	21527	11116	8249	23160	13868
1662	3810	19690	6122	13816	22787	11914	19008	20569	3526	3264	317	24865	2271	1060	6403	6594	17042	10317	22717	1897	4129	19064	16059	22376
24740	12969	12101	10512	11923	23213	13605	689	21456	1665	3344	1078	23985	14953	15794	21908	16151	8652	12008	24145	6435	7678	3525	4040	4586
2133	18833	19585	17326	20040	14056	13857	9985	3256	18568	13397	2338	19776	852	18259	20591	13924	20083	12118	5471	7409	21018	9508	13309	15684
3651	24388	22196	12654	7926	13579	8456	439	9541	12548	18291	12103	15813	11942	3375	5459	17924	466	3603	4895	19767	21039	11259	13472	15480
7661	3633	3967	1212	12642	756	14480	8266	23678	15050	19661	20573	18399	11199	14257	10605	9735	596	3238	19650	20768	1719	12933	11614	12539
6884	10795	7037	10634	2964	8607	2681	22182	14035	18521	12689	10478	10576	14645	1501	6835	12340	2779	15679	1847	18049	5574	21023	20166	340
11498	873	9703	20025	14243	14566	13958	10089	2139	9138	24083	2046	12471	3294	11021	23257	5816	14849	14044	13356	10624	8405	23283	7424	13556
7252	4740	10434	21661	6516	24936	11707	5092	18237	6408	20123	12000	5952	17922	5742	16966	21215	15074	17935	7020	8042	5971	10939	10363	17633
1243	5231	15296	4423	22091	13622	8480	24184	7305	22685	7856	6269	15920	14661	6105	19081	15171	21145	11897	19136	12048	2728	5384	10752	8178
20026	6173	12055	24638	1755	2099	8231	9528	5392	7650	16868	14765	14770	21059	13831	10914	14517	20263	6069	20226	2183	16701	14957	21833	18446
17849	16732	22280	23678	23369	235	13309	15129	2089	9590	14191	6592	10074	16986	24502	6295	7251	23055	3901	2349	23833	16491	754	3968	428
10194	10384	1186	3042	1419	13586	15786	4449	21854	23846	10046	6099	4667	5368	7537	24000	9303	9487	4488	21378	23574	22597	12389	21656	9920
13084	1398	17672	11171	21443	6399	1178	13906	18536	18635	17782	8727	16941	19849	6822	22191	1167	20450	10763	7105	21616	18529	16551	13604	13906
8609	10069	5055	4361	5473	24250	3358	24266	8672	18761	5906	1822	4007	6288	2497	3422	15376	2569	21829	4743	6277	12381	9849	7072	23021
15728	9544	18471	16719	12294	22007	12630	20553	8300	14861	1563	693	6998	22964	20568	23310	14823	13393	8339	24831	1412	21800	18368	6760	1166
1899	5767	17490	24260	10379	2209	772	15941	15566	3875	12817	15913	8253	6116	23573	8572	14265	10684	9309	15764	2453	2689	1958	22722	6540
1579	22847	14076	23537	3707	15398	9206	2016	2011	12114	6333	17327	5819	6946	23744	12733	10311	24933	10452	10732	6813	4098	16359	13992	7991
18917	12978	9601	23745	13416	1020	5300	22537	17033	13948	21270	9977	23480	5035	18236	1314	20226	1961	15231	10543	5625	990	15453	10064	1940
17833	3875	15516	18642	22828	7229	11095	6015	12805	19303	3342	7710	16850	11747	4567	5203	12501	11916	4176	21997	3519	12435	21582	15985	490
9411	1703	3532	24198	4929	19894	2247	18746	12348	2998	7808	13110	19178	17864	13456	19928	12991	20732	5884	11775	18406	15083	21689	20843	20473
21003	5495	512	9360	7644	14800	202	18048	15289	2475	10341	15890	11795	11923	286	6177	2729	4098	18292	18284	23794	11664	21688	14336	7374
5344	17006	3728	3765	9276	18732	13752	3048	11106	2159	1315	4217	5658	7046	22670	24767	9624	13471	11433	17383	17978	21043	24985	11264	16397
12174	1680	16700	17232	19310	19450	4337	22031	7998	8901	19459	3164	1671	20891	17017	20162	10817	11282	23205	6794	8244	10789	20402	8936	21862

Table A.1: Sample data for controlled testing of a districting problem using a 25 x 25 grid with randomly generated populations values in the range $(200, 25000)$ with a minimum value of 202, a maximum value of 24,985, a mean of 14,317.7, and a total population of 7,516,798.

Percentage of Infeasible Solutions for a 25x25 Grid Districting Problem								
Generation	μ=10		μ=50		μ=100		μ=500	
	Count	Percent	Count	Percent	Count	Percent	Count	Percent
10	6	0.6	26	0.52	42	0.42	341	0.682
20	0	0.0	20	0.40	35	0.35	256	0.512
30	3	0.3	19	0.38	46	0.46	220	0.440
40	2	0.2	17	0.34	45	0.45	221	0.442
50	6	0.6	18	0.36	38	0.38	180	0.360
60	5	0.5	23	0.46	38	0.38	185	0.370
70	0	0.0	21	0.42	42	0.42	163	0.326
80	7	0.7	21	0.42	40	0.40	178	0.356
90	2	0.2	21	0.42	43	0.43	195	0.390
100	5	0.5	14	0.28	36	0.35	191	0.382

Table A.2: Variations in the number of infeasible (i.e. non-contiguous) solutions in the population over time for typical run of an evolutionary algorithm operating on a 25x25 zone districting problem partitioned into four districts with fixed centers and using single-point crossover, no mutation, tournament size $k = 4$, and population sizes of $\mu = 10$, $\mu = 50$, $\mu = 100$, and $\mu = 500$.

	Contiguous Districting Problem with Random Mutation x 100 Trials											
	Population Size = 50				Population Size = 100				Population Size = 500			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
1	16587	122341.68	7442.93	0.6356	18319	12418.65	5056.42	0.6439	11815	123219.79	2234.75	0.6389
10	17619	88088.54	13452.49	0.3284	18319	83085.36	10506.87	0.3327	6831	78162.08	5638.39	0.3554
20	17619	74166.72	12911.56	0.2530	15600	72014.15	10652.23	0.3515	6831	64847.86	6369.11	0.4093
30	11567	67659.50	11994.88	0.2304	7910	67364.67	10554.08	0.3391	2649	58573.47	7369.37	0.4345
40	11475	65257.95	11047.71	0.2074	4071	64113.27	9251.85	0.3247	2549	53676.12	7457.31	0.4366
50	9319	62989.23	10325.32	0.2208	4071	60726.46	8961.54	0.3396	2455	48844.17	6723.37	0.4370
60	4578	62571.89	10264.12	0.1940	4438	59440.70	9209.70	0.3348	1395	45096.18	5863.90	0.4368
70	4578	61713.24	9785.04	0.1874	2850	57322.95	8641.18	0.3337	1395	42274.70	5073.35	0.4356
80	4578	60444.62	10472.03	0.1994	2850	55016.50	8498.68	0.3370	787	40378.43	3993.00	0.4402
90	2307	59775.23	10386.94	0.1952	2850	53245.20	8813.32	0.3385	943	39000.51	3628.80	0.4411
100	2630	59093.10	9119.78	0.1898	2850	51216.90	8865.60	0.3464	576	38162.43	3280.02	0.4417
110	2630	57878.09	7972.53	0.1914	2564	80910.77	8840.99	0.3286	576	37748.83	2808.67	0.4382
120	2630	57406.94	7892.12	0.1844	2518	49189.47	8925.08	0.3411	373	37358.65	2870.43	0.4391
130	2630	56081.95	8456.41	0.1940	1407	48048.93	8935.87	0.3394	373	37147.92	2945.41	0.4372
140	2630	56437.70	9143.10	0.1764	1407	47271.89	8533.17	0.3381	373	36936.09	2762.85	0.4380
150	2630	55607.37	7990.70	0.1816	1082	45756.60	8516.40	0.3515	373	36455.66	2885.57	0.4430
160	2630	54776.31	8961.02	0.1854	1407	45878.31	8557.46	0.3371	253	36467.16	2990.44	0.4419
170	1444	54631.42	7539.07	0.1788	1023	45534.89	7992.37	0.3309	253	36175.19	2950.58	0.4451
180	1444	53696.89	8589.27	0.1860	1023	44839.40	8750.50	0.3390	164	36677.20	3075.60	0.4362
190	1444	54252.68	7933.91	0.1722	1023	44351.93	9257.89	0.3399	164	36142.93	2941.02	0.4429
200	1444	53267.75	8681.25	0.1794	1023	44053.49	8449.42	0.3374	164	36333.55	2822.42	0.4375
210	1444	53341.87	7573.00	0.1710	1023	44173.88	8826.48	0.3333	164	35895.66	2895.58	0.4449
220	1444	53258.18	7367.04	0.1746	863	43648.67	8617.51	0.3338	164	36182.03	2784.57	0.4402
230	1444	52409.01	8795.91	0.1878	863	43061.68	8612.84	0.3389	164	36427.68	2943.35	0.4358
240	1444	51709.28	8507.28	0.1924	863	42877.43	9018.76	0.3362	100	35902.19	2936.11	0.4425
250	1444	51809.75	8132.93	0.1840	863	43191.01	8685.13	0.3341	100	36070.65	2969.88	0.4393
260	1444	51411.48	8249.19	0.1832	863	42838.97	9085.77	0.3361	100	35901.18	2894.37	0.4415
270	1444	51591.44	8654.53	0.1784	653	42316.09	8782.18	0.3415	100	36028.93	2861.25	0.4395
280	1444	51860.77	8624.80	0.1746	653	42632.90	9410.02	0.3349	100	36415.51	2917.85	0.4347
290	1444	51337.99	5861.44	0.1792	653	42217.40	9108.07	0.3377	100	36479.14	2809.53	0.4327
300	1444	51126.23	8768.07	0.1814	653	42455.60	8853.56	0.3358	100	35866.66	3015.24	0.4428

Table A.3: A comparison of candidate solution fitness for 100 trials of a genetic algorithm running for 300 generations with population sizes of $\mu = 50$, $\mu = 100$, and $\mu = 500$ for a district partitioning problem with a penalizing contiguity constraint on a 25x25 grid with no crossover operation, random mutation with probability $p_m = 0.001$, and no contiguity repair operation.

Generation	Contiguous Districting Problem with Neighborhood Mutation x 100 Trials											
	Population Size = 50				Population Size = 100				Population Size = 500			
	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
1	18306	106666.04	6656.79	0.9964	10928	104980.68	4591.99	0.9959	10063	104778.49	2057.08	0.9963
10	9369	49022.24	13608.61	0.9962	6708	39456.76	11393.55	0.9955	2183	28673.62	7139.17	0.9955
20	2835	34258.10	13635.55	0.9962	2352	23680.21	10209.70	0.9951	398	11073.76	4351.00	0.9952
30	2381	23267.89	12137.57	0.9948	822	13343.36	7940.63	0.9938	321	5032.19	1942.16	0.9956
40	1241	15885.33	10264.09	0.9946	74	7484.30	4265.46	0.9936	222	3129.86	1195.86	0.9952
50	350	11390.91	8554.26	0.9958	74	4987.11	2473.68	0.9937	117	2272.89	810.96	0.9952
60	350	8474.92	7121.59	0.9958	74	3696.98	1571.73	0.9946	62	1985.57	620.44	0.9947
70	350	6540.81	5192.21	0.9938	74	3248.46	1464.45	0.9926	62	1778.35	568.43	0.9947
80	289	5059.76	3331.26	0.9956	74	2738.14	1088.72	0.9946	62	1661.69	506.11	0.9948
90	289	4163.99	2541.72	0.9952	74	2481.17	1046.57	0.9948	62	1651.22	464.67	0.9947
100	289	3783.49	1719.42	0.9936	74	2469.46	1087.96	0.9934	62	1612.09	485.60	0.9950
110	166	3359.79	1337.23	0.9930	74	2275.24	943.59	0.9947	62	1716.10	523.20	0.9937
120	166	3209.21	1365.87	0.9908	74	2179.27	987.72	0.9942	62	1671.23	537.81	0.9943
130	166	2909.90	1259.38	0.9918	74	2228.80	961.31	0.9919	62	1614.27	492.59	0.9943
140	166	2663.66	1067.83	0.9932	74	2137.81	1007.76	0.9925	62	1569.48	449.15	0.9948
150	166	2625.45	1214.51	0.9924	74	2078.60	902.57	0.9925	62	1554.28	427.25	0.9951
160	166	2477.55	1178.93	0.9930	74	2044.78	888.97	0.9936	62	1614.67	462.98	0.9940
170	166	2344.55	1062.57	0.9944	74	2069.80	757.57	0.9923	62	1553.19	428.54	0.9945
180	166	2363.76	1116.30	0.9942	74	1968.87	783.41	0.9940	60	1611.12	475.10	0.9938
190	122	2250.57	1151.20	0.9940	74	1978.89	801.83	0.9927	60	1621.33	437.46	0.9937
200	122	2137.33	1052.70	0.9950	74	2026.48	803.92	0.9926	60	1555.62	398.65	0.9945
210	122	2293.33	1047.90	0.9930	74	1953.31	842.29	0.9929	60	1607.04	404.64	0.9938
220	122	2183.95	1075.52	0.9938	74	1793.52	760.59	0.9952	60	1608.40	449.17	0.9942
230	122	2423.31	1179.01	0.9914	74	1907.17	857.43	0.9935	60	1590.04	460.52	0.9941
240	122	2294.93	1104.22	0.9906	42	1862.86	723.83	0.9941	60	1596.46	437.43	0.9941
250	122	2170.59	1124.38	0.9918	42	1928.22	788.15	0.9930	60	1595.28	440.92	0.9942
260	122	2297.49	1297.98	0.9914	42	1913.17	790.43	0.9925	60	1531.19	407.65	0.9940
270	122	2135.77	1101.79	0.9924	42	1927.64	872.58	0.9932	60	1588.19	424.78	0.9940
280	122	2148.51	961.51	0.9936	42	1993.93	879.87	0.9922	60	1558.17	419.04	0.9944
290	122	2303.12	1115.44	0.9908	42	1901.54	760.44	0.9926	60	1586.46	411.46	0.9938
300	122	2129.75	1022.29	0.9918	42	1856.08	763.00	0.9934	60	1547.46	405.06	0.9944

Table A.4: A comparison of candidate solution fitness for 100 trials of a genetic algorithm running for 300 generations with population sizes of $\mu = 50$, $\mu = 100$, and $\mu = 500$ for a district partitioning problem with a penalizing contiguity constraint on a 25x25 grid with no crossover operation, implementation of the Neighborhood Mutation algorithm (see Algorithm 4.6) with probability $p_m = 0.001$, and no contiguity repair operation.

	Contiguous Districting Problem with Random Mutation and Local Repair x 100 Trials											
	Population Size = 50				Population Size = 100				Population Size = 500			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
1	2225	104098.33	7227.07	1.0000	5234	105791.30	4605.59	0.9992	2909	104323.99	1800.19	0.9991
10	1713	45537.90	13712.05	0.9988	2123	39890.88	12331.75	0.9994	1461	29228.67	7870.99	0.9995
20	1713	32205.27	13827.09	0.9996	2123	23555.49	10572.82	0.9993	580	12213.20	5360.47	0.9991
30	1361	22617.69	11991.79	0.9996	1134	14302.05	8299.38	0.9995	344	5248.97	2378.45	0.9993
40	1278	15534.91	10873.09	0.9988	1134	8406.03	5372.72	0.9995	344	3088.83	1097.51	0.9991
50	1278	10895.00	8809.29	0.9988	811	6016.07	3473.77	0.9990	206	2303.94	886.35	0.9994
60	993	8253.06	6874.36	0.9996	669	4428.84	2555.37	0.9991	206	1935.82	775.30	0.9991
70	221	5951.11	4190.41	0.9990	576	3691.69	2096.60	0.9989	174	1704.39	622.72	0.9992
80	221	4708.15	2772.70	0.9994	576	3240.52	1705.60	0.9992	110	1587.35	537.95	0.9990
90	221	4294.34	2394.03	0.9988	327	2905.89	1402.92	0.9989	110	1495.85	522.19	0.9991
100	221	3591.76	1935.72	0.9994	327	2559.53	1217.77	0.9994	110	1450.62	518.69	0.9990
110	221	3398.81	1853.22	0.9992	49	2331.99	1062.81	0.9989	66	1459.99	500.55	0.9990
120	221	3054.88	1567.88	0.9992	49	2174.23	1106.58	0.9989	66	1426.02	499.26	0.9991
130	221	2712.24	1392.16	0.9994	49	1954.46	848.36	0.9992	66	1435.84	493.56	0.9991
140	221	2547.71	1297.65	0.9996	49	1898.05	884.05	0.9986	66	1428.46	527.60	0.9990
150	221	2492.26	1263.54	0.9988	49	1909.21	870.10	0.9986	66	1418.50	488.08	0.9990
160	221	2427.19	1268.25	0.9986	49	1809.26	854.06	0.9987	66	1398.67	471.96	0.9992
170	221	2414.64	1232.89	0.9986	49	1783.49	687.85	0.9988	66	1381.25	436.28	0.9993
180	221	2230.36	1207.28	0.9990	49	1717.02	717.80	0.9991	66	1389.39	468.48	0.9992
190	221	2113.26	1145.19	0.9988	49	1678.05	791.98	0.9984	66	1409.18	483.59	0.9991
200	221	2110.38	1138.17	0.9990	49	1705.26	770.68	0.9986	66	1378.70	478.73	0.9992
210	221	2087.16	1039.99	0.9986	49	1608.72	687.43	0.9991	66	1393.10	470.86	0.9991
220	221	2012.85	1079.12	0.9982	49	1551.68	694.81	0.9991	66	1391.77	486.47	0.9990
230	221	1913.75	948.11	0.9986	49	1622.49	712.76	0.9980	66	1362.43	481.12	0.9993
240	221	1840.35	988.45	0.9984	49	1601.79	682.17	0.9983	66	1419.59	502.52	0.9992
250	221	1921.73	880.05	0.9986	49	1583.18	741.63	0.9986	66	1394.08	464.73	0.9991
260	221	1757.59	775.27	0.9990	49	1575.56	640.26	0.9986	66	1398.95	475.81	0.9991
270	221	1714.35	749.78	0.9992	49	1508.73	631.56	0.9986	66	1374.02	473.04	0.9991
280	221	1732.36	827.06	0.9990	49	1509.96	665.52	0.9983	66	1364.19	505.53	0.9992
290	221	1710.53	708.84	0.9990	49	1473.06	696.51	0.9991	66	1642.68	483.93	0.9991
300	221	1679.60	819.10	0.9986	49	1532.43	681.88	0.9985	66	1369.70	482.63	0.9992

Table A.5: A comparison of candidate solution fitness for 100 trials of a genetic algorithm running for 300 generations with population sizes of $\mu = 50$, $\mu = 100$, and $\mu = 500$ for a district partitioning problem with a penalizing contiguity constraint on a 25x25 grid with no crossover operation, random mutation with probability $p_m = 0.001$, and implementation of the Local Repair algorithm (see Algorithm 4.7) on candidate solutions violating the contiguity constraint.

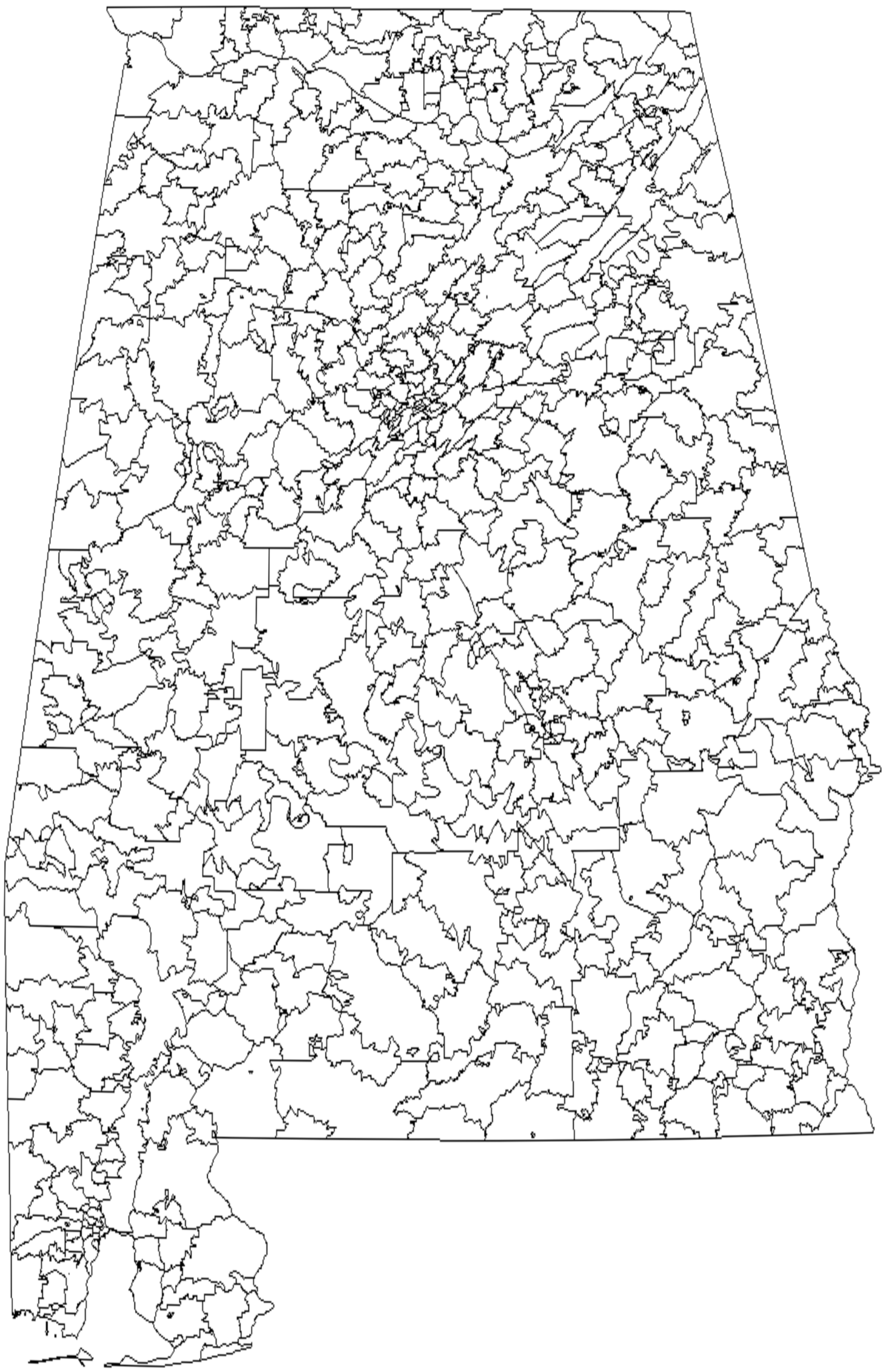

Figure A.1: All Zip Code Tabulation Areas (ZCTAs) contained within the state of Alabama. ZCTAs, which are geographic areas used in the decennial census, are approximations of U.S. Postal Service Zone Improvement Plan (ZIP) Code service areas [94][172][171]. There are 642 ZCTAs contained entirely within Alabama.

	Contiguous Districting Problem for the State of Alabama with a Population Equality Heuristic and Neighborhood Mutation x 100 Iterations							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	40929	83511.34	12959.52	0.9818	36090	82903.11	12161.67	0.9551
20	32028	73788.76	13799.14	0.9816	20569	69002.55	12867.06	0.9487
30	22448	66340.16	14428.06	0.9814	17204	59961.86	12579.68	0.9455
40	16480	60298.05	15421.83	0.9781	10569	82286.18	12733.68	0.9443
50	9290	55507.38	15174.74	0.9774	8525	46179.84	12589.78	0.9336
60	8297	50158.03	14568.64	0.9788	6271	40587.18	12453.84	0.9303
70	5040	46161.53	14451.34	0.9770	2160	35318.74	12218.52	0.9299
80	4566	42190.11	14416.92	0.9749	1421	30805.41	12163.36	0.9300
90	4222	38120.86	13510.26	0.9746	1232	26676.35	11496.63	0.9268
100	3898	35311.06	13566.66	0.9743	1164	23780.51	10847.69	0.9167
110	3031	32435.69	13059.01	0.9731	543	21106.65	9967.10	0.9159
120	1521	39792.30	13175.72	0.9693	192	18790.21	9646.02	0.9161
130	1003	27511.62	13241.20	0.9687	180	16840.20	8706.25	0.9112
140	618	25180.67	12887.79	0.9674	167	15442.31	8100.10	0.9097
150	382	23157.07	12503.57	0.9649	136	13934.58	7144.04	0.9082
160	382	21313.57	11829.88	0.9660	98	13092.59	6309.20	0.9033
170	314	19249.98	11106.94	0.9683	93	12325.90	5839.03	0.9015
180	314	17347.83	10365.13	0.9685	93	11615.76	5349.78	0.9035
190	314	16186.55	10397.28	0.9625	93	10636.49	4417.85	0.9090
200	314	14770.97	10216.34	0.9636	93	10346.40	3699.27	0.9030
210	314	13206.81	9971.11	0.9647	93	10252.74	3643.61	0.8983
220	314	11875.26	8971.42	0.9654	93	9894.39	3455.70	0.8991
230	314	10870.07	8316.52	0.9637	93	9548.26	2965.60	0.8991
240	118	10181.98	8096.41	0.9637	93	9081.32	2894.39	0.9034
250	104	9258.44	7878.26	0.9656	93	9075.40	2699.41	0.9033
260	104	8704.52	7439.97	0.9622	93	8899.22	2547.33	0.9030
270	104	8287.07	7073.91	0.9595	93	9340.61	2770.75	0.8946
280	102	7820.23	6756.55	0.9591	93	8798.67	2492.36	0.9009
290	97	7423.68	6227.85	0.9587	93	8666.25	2074.94	0.9037
300	97	6795.71	6134.39	0.9623	93	8855.03	2114.57	0.8983
310	97	6765.19	5676.94	0.9583	93	8956.71	2191.96	0.8957
320	97	6356.73	5550.00	0.9602	93	9145.62	2500.55	0.8924
330	97	6407.12	5316.02	0.9552	93	8931.67	2363.12	0.8963
340	87	5675.25	5257.80	0.9630	68	9081.35	2222.95	0.8923
350	87	5620.56	4881.41	0.9612	68	8762.01	2104.30	0.8964
360	72	5519.81	5271.75	0.9600	66	8428.93	2280.58	0.9005
370	72	5278.38	4210.58	0.9601	66	8641.75	2133.85	0.8990
380	72	5258.99	3968.57	0.9594	66	8825.56	2403.72	0.8953
390	72	5375.11	3856.55	0.9539	66	8467.42	2289.39	0.9007
400	72	4868.05	3620.26	0.9622	66	8662.80	2265.97	0.8987
410	72	4842.53	3688.40	0.9601	66	8473.66	2080.94	0.9008
420	72	4673.13	3067.90	0.9600	66	8428.47	1993.42	0.8998
430	72	4558.00	2305.39	0.9590	66	8251.90	2009.00	0.9025
440	72	4464.89	2232.71	0.9591	66	8682.62	2171.61	0.8957
450	72	4331.62	1880.35	0.9585	66	8534.47	2287.96	0.8975
460	72	4399.35	1762.86	0.9564	66	8592.38	2079.97	0.8982
470	72	4337.46	1723.03	0.9659	66	8866.99	1925.97	0.8941
480	72	4221.92	1484.98	0.9592	66	8805.85	2086.17	0.8956
490	72	4165.95	1800.44	0.9596	66	8503.00	1991.27	0.8993
500	72	4201.23	1592.21	0.9568	66	9055.97	2120.04	0.8898

Table A.6: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with a equal population distribution heuristic, population size $\mu = 100$, no crossover, and Neighborhood Mutation with mutation rates of $p_m = 0.001$ and $p_m = 0.0025$.

	Contiguous Districting Problem for the State of Alabama with a Population Equality Heuristic and Neighborhood Mutation x 100 Iterations							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	36866	80740.45	9948.44	0.9039	38830	84958.80	10203.57	0.8011
20	28108	66064.75	10910.47	0.8932	17115	68120.56	10894.40	0.7631
30	13048	56339.75	12231.85	0.8792	9792	58920.77	11891.33	0.7238
40	6355	48843.91	12023.41	0.8668	5432	52264.14	10809.65	0.6974
50	2410	42518.77	11903.30	0.8558	3841	47236.80	10027.34	0.6706
60	1583	37497.09	10696.35	0.8407	3282	43877.04	8740.69	0.6554
70	805	33193.77	10074.43	0.8333	1516	51155.33	8524.77	0.6250
80	677	28934.00	9407.26	0.8300	1335	41559.69	7803.01	0.6025
90	629	25763.75	8125.57	0.8213	2478	40663.90	7310.08	0.5909
100	607	24073.33	7906.16	0.8081	1968	41255.92	7047.34	0.5711
110	592	22348.03	6544.96	0.8075	1923	42239.86	6974.03	0.5460
120	552	21275.78	5950.20	0.8019	1847	51016.06	7181.23	0.5579
130	325	20355.88	5439.78	0.8013	1476	43393.21	7206.31	0.5241
140	242	19600.63	5152.64	0.7970	1424	42912.46	6843.66	0.5242
150	225	19221.42	4298.50	0.7949	1215	43696.79	7211.48	0.5142
160	147	18389.83	3756.03	0.8007	986	43990.88	7661.08	0.5161
170	183	18692.80	3860.59	0.7871	1586	46336.74	7760.98	0.4931
180	163	18450.46	3404.05	0.7881	1292	45773.39	8151.39	0.4881
190	163	17756.65	3563.29	0.7955	1586	48024.01	8513.34	0.4620
200	147	18027.82	3681.99	0.7894	1164	47686.58	8573.98	0.4654
210	147	18340.50	3692.69	0.7833	1407	48529.72	9449.94	0.4599
220	147	18019.95	3131.00	0.7838	1637	48997.91	8598.36	0.4473
230	147	17747.28	3385.36	0.7868	1838	50065.41	8661.80	0.4318
240	147	18450.57	3234.70	0.7781	2062	51230.34	8988.07	0.4193
250	147	18202.87	3263.53	0.7795	2129	51730.13	9280.16	0.4125
260	147	18036.73	2895.16	0.7827	1709	53069.78	9508.16	0.3965
270	147	17767.93	2769.06	0.7872	1785	53229.10	9916.93	0.3966
280	147	18060.75	3234.39	0.7804	1986	54908.51	9415.03	0.3749
290	117	17996.26	3134.62	0.7828	1793	55414.56	9388.30	0.3656
300	117	18355.05	3064.82	0.7757	1957	56298.37	9574.67	0.3501
310	117	18126.80	3277.13	0.7802	1957	57014.41	10376.71	0.3465
320	111	17807.45	3037.38	0.7853	2410	58222.09	9883.22	0.3251
330	111	17941.45	3371.47	0.7811	2151	58958.27	10156.95	0.3192
340	78	17863.22	3301.50	0.7815	2593	58780.44	10022.22	0.3056
350	78	18186.26	3525.83	0.7776	1758	59843.18	9752.58	0.2876
360	78	18061.90	3394.52	0.7772	2899	59969.37	9496.87	0.2935
370	78	18144.19	2913.68	0.7757	1796	60527.99	9636.52	0.2814
380	78	18078.51	3240.72	0.7787	2149	61483.98	9403.79	0.2532
390	78	18338.90	3260.04	0.7753	2797	61559.64	6134.15	0.2594
400	78	17919.40	3147.58	0.7815	2425	63709.28	9083.90	0.2470
410	78	18256.64	3343.79	0.7739	2982	63172.32	9255.22	0.2327
420	78	17817.04	3304.31	0.7859	3556	63406.32	8041.34	0.2228
430	78	18097.27	3026.94	0.7808	3806	65231.82	7889.00	0.2010
440	78	17757.38	3514.09	0.7821	4406	64394.70	8076.03	0.1990
450	78	18028.62	3453.45	0.7788	5043	64416.58	7108.46	0.1826
460	78	17929.08	3226.17	0.7800	4180	65196.99	6929.46	0.1665
470	78	17487.05	3086.09	0.7871	2964	66216.90	6450.17	0.1627
480	78	17871.46	3287.83	0.7835	4582	65691.52	6525.76	0.1612
490	78	18227.30	3571.47	0.7757	2345	66589.67	6666.32	0.1450
500	78	18239.60	3384.97	0.7787	2524	66786.85	6682.30	0.1351

Table A.7: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with a equal population distribution heuristic, population size $\mu = 100$, no crossover, and Neighborhood Mutation with mutation rates of $p_m = 0.005$ and $p_m = 0.010$.

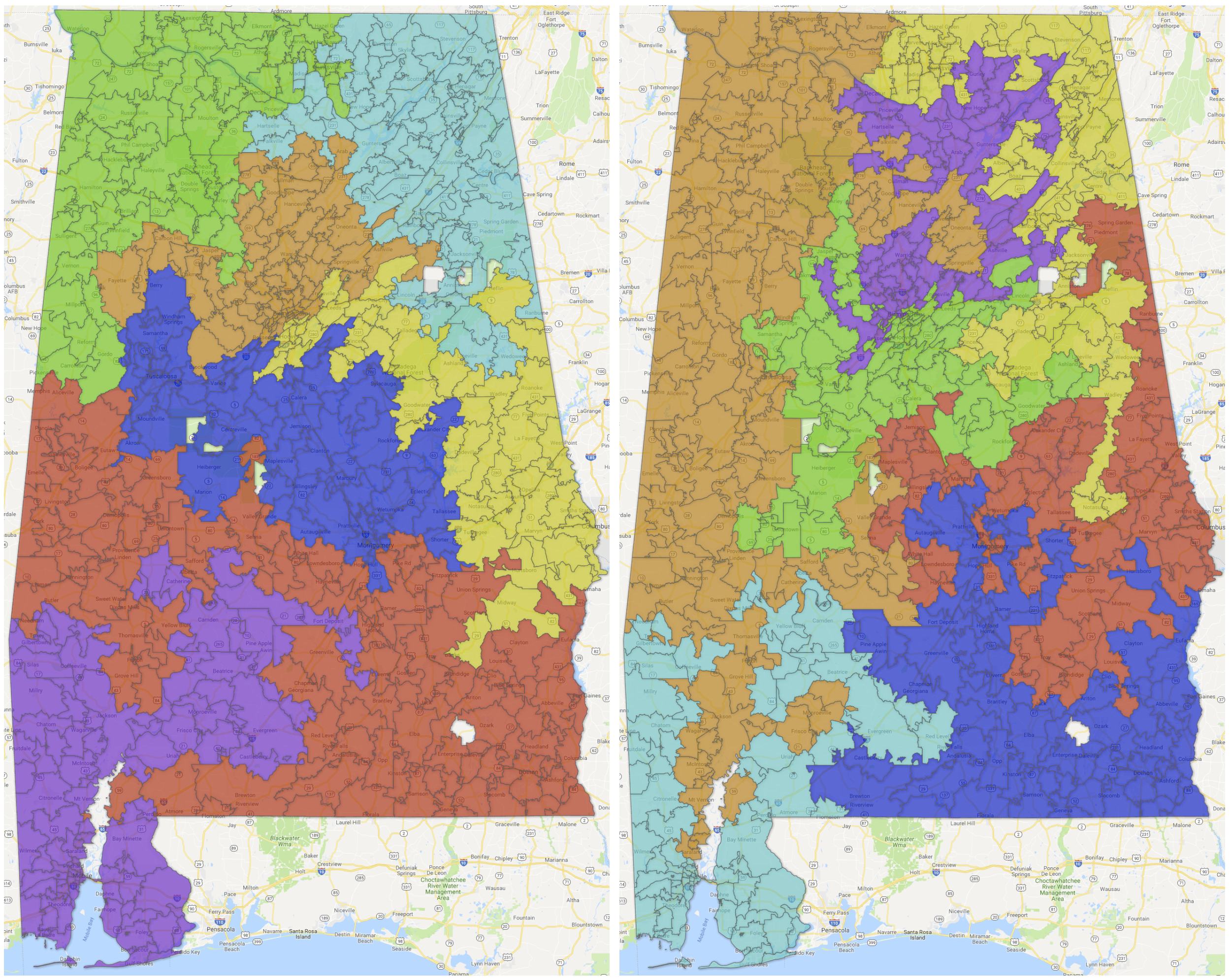

Figure A.2: Two near-optimal solutions for districting the state of Alabama into seven contiguous zones based on equal distribution of total population, with randomly generated zone centers. The maps depict, from left, a distribution of total population with a difference of no more than 85 and 119 persons, respectively, between any two zones.

	Contiguous Districting Problem for the State of Alabama with a Minority Population Heuristic, Neighborhood Mutation, and Local Repair x 100 Iterations							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	34245	54018.85	5806.17	0.9942	37662	52409.13	5522.58	0.9861
20	31482	49409.56	6071.81	0.9932	34587	47029.48	5528.32	0.9844
30	31301	46776.40	6088.53	0.9926	30985	43703.61	5512.66	0.9803
40	29303	44658.49	5981.08	0.9936	27181	40874.46	5666.16	0.9763
50	27470	42804.79	5923.37	0.9929	24565	38294.88	6161.87	0.9779
60	26129	41105.97	5989.52	0.9916	22539	35950.44	6172.83	0.9712
70	24986	39516.43	6264.79	0.9905	20307	33777.87	6337.29	0.9687
80	21549	38150.83	6292.52	0.9900	15077	31552.21	6498.07	0.9678
90	19420	36711.13	6314.45	0.9897	12595	29597.42	6551.52	0.9662
100	17778	35431.69	6182.50	0.9875	7880	27618.98	6371.76	0.9620
110	15619	34003.53	6297.52	0.9881	6426	25776.85	6600.21	0.9631
120	13890	32864.37	6448.50	0.9880	5694	24104.52	6726.95	0.9610
130	11093	31635.03	6624.78	0.9891	4954	22490.25	6882.53	0.9559
140	10870	30343.21	6680.31	0.9873	4076	21059.59	6991.87	0.9496
150	9348	29436.61	6864.39	0.9845	3524	19672.62	6857.18	0.9480
160	8778	28266.04	6956.46	0.9865	1710	18157.40	6671.52	0.9514
170	8521	27251.46	7043.26	0.9842	1423	16860.48	6635.54	0.9487
180	7973	26282.99	6895.47	0.9818	1299	15850.25	6704.04	0.9427
190	7840	25356.62	7070.98	0.9828	982	14690.13	6806.29	0.9418
200	6594	24598.43	7024.55	0.9830	654	13697.76	6693.04	0.9385
210	5528	23676.00	6905.40	0.9844	351	12791.98	6419.97	0.9371
220	5215	22639.97	6846.41	0.9832	248	11955.06	6451.41	0.9359
230	3439	21528.03	7046.93	0.9843	221	11118.95	6397.72	0.9355
240	2441	20662.70	7143.57	0.9844	163	10393.26	6121.72	0.9321
250	1277	19799.44	7255.59	0.9816	121	9749.30	5877.39	0.9297
260	1057	19080.38	7305.05	0.9802	107	8980.00	5574.33	0.9310
270	895	18311.07	7384.58	0.9813	86	8332.11	5424.01	0.9294
280	669	17653.48	7427.65	0.9783	86	7677.50	5225.03	0.9351
290	539	16794.97	7539.36	0.9805	68	7435.38	5167.94	0.9248
300	461	16030.23	7500.60	0.9799	64	6885.36	4781.39	0.9263
310	426	15308.45	7491.54	0.9805	53	6478.49	4638.14	0.9236
320	234	14578.55	7492.57	0.9784	50	6103.36	4392.97	0.9239
330	158	13991.62	7533.53	0.9779	33	5759.24	4156.16	0.9234
340	158	13318.22	7549.93	0.9782	33	5409.69	4042.35	0.9237
350	158	12720.62	7617.66	0.9790	33	5274.83	3920.60	0.9197
360	158	12323.04	7569.38	0.9742	24	4987.48	3721.00	0.9242
370	142	11704.15	7477.50	0.9773	21	4739.06	3529.32	0.9256
380	142	11226.71	7442.16	0.9727	17	4529.30	3541.91	0.9229
390	95	10721.42	7416.83	0.9753	17	4466.32	3229.98	0.9228
400	72	10206.34	7469.35	0.9758	17	4340.10	3136.30	0.9199
410	52	9822.29	7453.45	0.9743	17	4295.94	2844.08	0.9172
420	51	9479.58	7390.95	0.9727	12	4103.55	2762.41	0.9184
430	39	9045.56	7494.67	0.9737	11	4155.60	2656.36	0.9155
440	33	8734.29	7447.61	0.9729	10	3879.40	2674.60	0.9195
450	30	8386.42	7431.60	0.9735	10	3873.83	2212.79	0.9171
460	30	8063.35	7291.43	0.9722	10	3722.83	2264.19	0.9175
470	30	7762.04	7255.33	0.9711	10	3680.40	2126.73	0.9169
480	30	7497.71	7157.98	0.9714	10	3407.87	1937.95	0.9230
490	30	7197.05	7083.83	0.9714	10	3487.26	1878.91	0.9181
500	30	6955.94	6973.43	0.9693	10	3438.57	1852.00	0.9190

Table A.8: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with an equal distribution of minority population heuristic, population size $\mu = 100$, no crossover, Neighborhood Mutation with mutation rates of $p_m = 0.005$ and $p_m = 0.010$, and Local Repair.

	Contiguous Districting Problem for the State of Alabama with a Minority Population Heuristic, Neighborhood Mutation, and Local Repair x 100 Iterations							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	34430	50565.89	5041.11	0.9711	29926	50978.25	5342.03	0.9323
20	30040	44707.08	5140.87	0.9596	25451	43664.78	4848.51	0.9029
30	22059	40763.20	5735.54	0.9506	22867	38699.24	4838.21	0.8860
40	16038	37368.23	6109.45	0.9451	17937	34719.89	4927.39	0.8631
50	13757	34415.35	6492.85	0.9326	12731	31223.49	5256.42	0.8363
60	12605	31369.68	6720.87	0.9285	6976	28198.42	5412.54	0.8206
70	7778	28677.92	6799.87	0.9187	3131	25439.30	5760.86	0.8095
80	5109	26214.61	6976.58	0.9121	1168	23476.39	5339.47	0.7889
90	2853	23961.41	6809.63	0.9067	1129	21351.72	5346.44	0.7796
100	1408	21802.72	6759.18	0.9004	919	19996.94	5298.17	0.7548
110	1209	19844.18	6557.31	0.8940	482	18617.26	5259.85	0.7390
120	971	18076.02	6390.29	0.8866	493	17681.50	5564.48	0.7230
130	702	16563.39	6283.59	0.8835	302	16674.37	5265.03	0.7167
140	586	15354.96	6319.74	0.8728	342	16078.58	5094.31	0.7072
150	534	14148.23	6264.17	0.8750	559	15143.43	4695.86	0.7087
160	212	13265.05	5991.41	0.8715	450	14974.82	4620.18	0.6981
170	157	12427.51	5937.30	0.8646	399	14901.09	4622.46	0.6887
180	119	11674.17	5785.05	0.8628	462	14510.01	4345.84	0.6799
190	100	10896.33	5366.53	0.8633	425	14222.15	4050.94	0.6784
200	100	10480.38	5155.82	0.8514	520	14389.76	5176.68	0.6636
210	65	9960.25	5035.05	0.8519	275	14186.62	3631.50	0.6622
220	45	9362.77	4724.21	0.8448	224	14202.91	3613.29	0.6597
230	45	8952.46	4500.13	0.8461	336	13994.61	3408.16	0.6612
240	41	8381.39	4149.85	0.8484	356	13911.99	3163.02	0.6596
250	38	8066.84	4041.19	0.8468	402	13839.49	3240.18	0.6515
260	38	7845.75	3817.15	0.8422	236	13711.49	2897.80	0.6598
270	38	7468.45	3541.47	0.8449	437	13740.12	3083.68	0.6488
280	38	7368.10	3602.87	0.8362	265	13945.67	2993.88	0.6424
290	26	6946.94	3206.66	0.8418	293	13922.52	2907.75	0.6428
300	26	6845.20	2907.62	0.8360	197	14243.65	2774.20	0.6265
310	26	6682.12	2713.29	0.8400	522	13825.06	2617.02	0.6379
320	26	6593.99	2662.80	0.8368	331	14147.33	2638.30	0.6276
330	26	3589.24	2443.43	0.8347	487	14257.11	2724.95	0.6256
340	26	6535.42	2431.89	0.8305	309	14201.28	2567.80	0.6307
350	26	6542.88	2300.81	0.8294	443	14115.21	2699.67	0.6287
360	26	6523.78	2212.61	0.8265	389	14440.86	2786.27	0.6160
370	26	6454.56	2068.77	0.8301	445	14508.00	2720.42	0.6148
380	25	6491.26	2049.75	0.8239	330	14578.94	2651.09	0.6215
390	26	6316.42	1965.85	0.8304	385	14561.90	2510.33	0.6130
400	25	6435.62	1949.88	0.8258	403	14543.19	2515.72	0.6142
410	24	6193.65	1685.54	0.8273	340	15026.13	2240.06	0.5934
420	25	6291.97	1815.27	0.8258	592	14756.26	2569.50	0.6019
430	25	6458.26	1706.85	0.8200	479	14812.96	2626.19	0.6049
440	25	6398.85	1856.28	0.8211	414	14998.51	2449.94	0.6035
450	17	6127.55	1762.69	0.8346	366	15039.35	2635.99	0.5969
460	24	6257.64	1513.49	0.8285	459	15248.59	2392.17	0.5869
470	24	6292.15	1344.69	0.8238	490	15178.72	2678.12	0.5927
480	24	6195.47	1539.71	0.8256	507	15166.08	2513.10	0.5945
490	24	6169.20	1578.59	0.8267	468	15385.07	2468.87	0.5883
500	24	6094.29	1601.75	0.8313	509	15439.74	2365.82	0.5841

Table A.9: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with an equal distribution of minority population heuristic, population size $\mu = 100$, no crossover, Neighborhood Mutation with mutation rates of $p_m = 0.005$ and $p_m = 0.010$, and Local Repair.

	Contiguous Districting Problem for the State of Alabama with a District Compactness Heuristic and Neighborhood Mutation x 100 Iterations							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	145.3	316.90	71.096	0.9841	154.7	313.65	68.81	0.9608
20	143.6	314.96	70.947	0.9817	151.9	311.43	69.09	0.9517
30	141.1	312.89	70.622	0.9814	149.3	309.02	68.45	0.9462
40	139.7	311.26	70.410	0.9785	147.2	306.80	68.19	0.9398
50	138.1	309.37	70.440	0.9774	145.4	304.72	68.22	0.9338
60	135.1	307.53	70.192	0.9770	142.7	302.52	69.82	0.9304
70	132.8	305.62	69.798	0.9783	141.7	300.85	67.22	0.9238
80	130.3	304.25	69.948	0.9734	139.5	299.13	66.92	0.9181
90	129.1	302.49	69.640	0.9731	138.4	297.18	66.82	0.9156
100	127.9	301.08	68.860	0.9704	136.6	295.53	67.02	0.9113
110	126.9	299.60	68.839	0.9687	135.7	293.72	66.57	0.9093
120	125.5	297.98	68.655	0.9694	134.6	291.76	65.82	0.9087
130	125.0	296.36	68.109	0.9699	133.4	290.53	65.81	0.9015
140	123.8	294.94	68.074	0.9684	132.5	289.36	66.22	0.8942
150	122.9	293.79	68.026	0.9637	131.5	287.29	64.73	0.8967
160	121.4	292.29	67.244	0.9641	130.0	286.48	65.05	0.8862
170	120.5	290.91	66.953	0.9633	129.5	284.77	65.99	0.8861
180	119.1	289.43	66.840	0.9649	128.3	282.88	63.92	0.8881
190	118.1	288.11	66.579	0.9656	127.2	281.58	64.47	0.8843
200	117.6	287.45	66.260	0.6598	125.9	280.91	63.52	0.8753
210	116.8	286.22	65.927	0.9608	124.3	279.43	64.22	0.8755
220	116.1	285.06	65.552	0.9604	123.3	278.34	64.14	0.8719
230	115.7	284.33	65.299	0.9563	123.1	277.17	63.02	0.8694
240	114.9	283.23	36.834	0.9556	121.5	276.08	63.95	0.8667
250	114.7	282.04	64.957	0.9560	120.7	275.50	63.21	0.8598
260	114.4	280.83	64.679	0.9579	120.0	274.09	63.38	0.8615
270	114.3	279.86	64.227	0.9568	119.9	272.77	63.23	0.8628
280	113.9	279.14	64.105	0.9539	119.2	271.90	62.50	0.8595
290	113.6	278.20	63.726	0.9533	117.9	270.40	62.47	0.8628
300	113.3	277.02	63.240	0.9554	117.5	270.50	62.47	0.8497
310	112.8	276.13	63.401	0.9543	116.8	268.81	32.11	0.8555
320	112.8	275.07	62.892	0.9546	116.4	268.06	62.45	0.8514
330	112.8	274.45	62.677	0.9516	115.6	266.89	62.31	0.8520
340	112.8	273.13	62.111	0.9551	114.8	267.36	62.09	0.8369
350	112.4	272.80	61.768	0.9491	114.3	365.98	61.51	0.8405
360	112.0	271.78	61.422	0.9499	113.9	264.75	61.61	0.8422
370	111.8	271.36	61.229	0.9451	113.4	264.33	61.36	0.8356
380	110.6	269.81	61.093	0.9511	113.0	263.45	61.31	0.8347
390	110.2	269.10	60.602	0.9488	112.3	262.09	60.87	0.8381
400	109.1	268.46	60.745	0.9465	111.8	261.41	60.68	0.8354
410	108.5	267.68	60.117	0.9449	111.3	259.71	60.72	0.8420
420	108.2	266.55	60.291	0.9478	110.7	259.68	60.51	0.8322
430	107.1	265.66	59.757	0.9481	110.2	258.86	60.72	0.8303
440	106.5	265.32	59.331	0.9432	109.4	258.58	60.64	0.8243
450	106.5	264.85	59.477	0.9392	109.3	256.84	60.54	0.8324
460	106.1	263.79	59.189	0.9415	108.7	256.22	59.17	0.8291
470	106.0	263.16	59.975	0.9401	108.5	255.57	59.77	0.8267
480	105.8	261.75	58.686	0.9466	107.7	255.22	59.32	0.8218
490	105.5	261.57	28.735	0.9414	107.3	253.86	59.61	0.8270
500	104.9	260.62	58.230	0.9437	107.0	253.52	59.76	0.8225

Table A.10: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with an average distance from center compactness heuristic with a population size $\mu = 100$, no crossover, and Neighborhood Mutation with mutation rates of $p_m = 0.001$ and $p_m = 0.0025$.

	Contiguous Districting Problem for the State of Alabama with an Average Distance Compactness Heuristic and Neighborhood Mutation x 100 Iterations							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	166.8	330.51	75.19	0.9081	123.1	330.26	74.94	0.8091
20	163.8	328.03	74.47	0.8960	121.5	331.27	75.21	0.7562
30	161.6	326.04	75.22	0.8792	120.3	330.83	75.28	0.7224
40	159.6	324.14	74.41	0.8636	119.2	330.44	74.60	0.6905
50	156.7	321.59	73.59	0.8538	118.3	331.00	75.56	0.6504
60	154.8	320.11	73.57	0.8351	117.3	332.11	73.92	0.6083
70	153.0	317.66	72.99	0.8291	116.5	331.71	74.50	0.5857
80	151.6	315.90	73.56	0.8174	116.2	333.72	74.15	0.5415
90	150.8	315.92	72.83	0.7889	115.5	333.75	72.87	0.5183
100	150.1	313.24	72.88	0.7895	115.0	335.59	73.08	0.4784
110	149.7	311.97	73.34	0.7768	114.8	335.61	72.48	0.4586
120	148.0	310.80	72.85	0.7646	114.6	336.15	73.21	0.4363
130	145.1	308.41	73.20	0.7664	114.4	337.37	72.47	0.4083
140	142.5	308.02	72.96	0.7495	114.2	337.65	72.83	0.3903
150	140.5	306.85	72.62	0.7407	113.7	337.70	73.15	0.3748
160	139.3	305.65	71.53	0.7338	113.5	339.37	71.31	0.3429
170	138.1	304.50	71.84	0.7268	113.2	339.10	72.39	0.3337
180	136.9	303.10	72.33	0.7237	113.2	338.51	72.41	0.3257
190	136.3	302.48	71.24	0.7133	113.0	338.97	72.47	0.3065
200	135.8	301.68	71.10	0.7054	113.0	337.76	70.40	0.3044
210	135.2	301.33	71.83	0.6935	112.9	338.87	71.09	0.2745
220	133.6	299.44	70.28	0.6984	112.9	339.38	69.76	0.2462
230	130.7	299.84	69.76	0.6802	112.9	337.31	69.32	0.2476
240	128.9	297.32	70.26	0.6912	112.9	336.45	68.88	0.2423
250	126.0	297.83	70.64	0.6727	112.9	336.14	68.40	0.2183
260	124.6	297.56	69.86	0.6629	112.9	334.36	66.87	0.2168
270	122.4	296.03	68.88	0.6654	112.9	333.88	66.65	0.1668
280	120.3	295.48	69.05	0.6590	112.9	333.37	66.43	0.1815
290	118.8	295.29	69.58	0.6493	112.9	331.76	66.25	0.1672
300	117.1	293.37	69.25	0.6571	112.9	330.21	65.02	0.1559
310	111.9	293.93	69.15	0.6401	112.9	328.20	66.04	0.1419
320	110.8	293.35	69.48	0.6350	112.9	326.09	65.05	0.1321
330	107.3	292.13	68.73	0.6371	112.9	324.75	64.38	0.1116
340	105.4	291.57	68.19	0.6321	112.9	322.53	64.87	0.1013
350	103.8	289.58	69.02	0.6420	112.9	320.27	64.99	0.0972
360	103.2	290.71	68.26	0.6210	112.9	318.00	63.97	0.0886
370	102.1	289.59	69.26	0.6225	112.9	315.73	63.50	0.0750
380	100.6	289.60	67.13	0.6133	112.9	313.48	63.27	0.0593
390	99.7	287.73	67.15	0.6229	112.9	309.77	62.54	0.0624
400	98.5	287.15	66.81	0.6202	112.9	307.82	62.17	0.0524
410	98.3	287.59	68.11	0.6074	112.9	305.52	62.11	0.0445
420	97.3	285.47	67.29	0.6198	112.9	302.61	61.28	0.0440
430	96.6	285.54	66.26	0.6108	112.9	300.23	60.39	0.0370
440	95.3	284.55	66.43	0.6126	112.9	297.08	59.92	0.0303
450	94.9	285.12	66.16	0.5981	112.9	294.54	58.92	0.0269
460	94.2	285.77	65.65	0.5841	113.0	292.00	58.60	0.0266
470	93.9	283.19	65.40	0.6026	113.0	289.08	57.50	0.0225
480	93.7	282.52	65.57	0.6018	113.0	286.10	56.52	0.0152
490	93.5	283.61	65.34	0.5838	113.0	283.26	55.96	0.0149
500	93.4	282.31	64.72	0.5894	112.9	280.64	54.82	0.0111

Table A.11: A comparison of candidate solution fitness, by generation, for 100 trials of an evolutionary algorithm for a contiguous districting problem for the State of Alabama with an average distance from center compactness heuristic with a population size $\mu = 100$, no crossover, and Neighborhood Mutation with mutation rates of $p_m = 0.005$ and $p_m = 0.010$.

	Contiguous Districting Problem for the State of Alabama with a District Compactness Heuristic, Neighborhood Mutation, and Local Repair x 100 Iterations							
	With Local Repair				Without Local Repair			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	169.4	327.30	62.197	0.9643	183.5	360.23	61.943	0.9064
20	167.2	324.43	62.479	0.9555	180.7	359.38	62.332	0.8938
30	162.4	320.77	61.651	0.9501	176.2	360.25	63.862	0.8749
40	156.9	317.45	60.838	0.9444	174.0	358.42	61.136	0.8665
50	149.9	314.64	61.500	0.9368	171.2	360.69	61.944	0.8431
60	144.7	313.51	61.113	0.9236	168.3	361.07	60.882	0.8288
70	138.6	310.11	59.950	0.9212	164.8	360.40	61.853	0.8188
80	134.0	308.38	61.689	0.9134	162.0	360.34	60.898	0.8070
90	127.9	306.39	59.724	0.9073	158.5	359.44	59.950	0.7989
100	123.0	307.46	61.170	0.8900	155.8	360.88	60.416	0.7817
110	116.6	304.56	59.811	0.8891	153.4	360.46	59.510	0.7728
120	112.7	304.09	58.316	0.8785	151.9	360.61	58.998	0.7625
130	108.9	301.82	58.422	0.8760	149.2	360.01	59.635	0.7556
140	106.5	300.86	57.094	0.8693	147.8	360.10	60.100	0.7466
150	105.6	298.57	57.055	0.8680	146.9	361.00	60.266	0.7347
160	104.3	297.76	58.664	0.8612	146.2	360.45	58.015	0.7291
170	103.7	294.35	57.727	0.8646	145.7	359.22	58.673	0.7267
180	102.9	292.44	56.581	0.8629	144.6	360.16	57.811	0.7158
190	101.9	292.54	58.385	0.8530	143.8	359.69	57.685	0.7109
200	100.6	291.28	57.164	0.8489	143.1	359.03	59.740	0.7064
210	99.8	289.43	56.540	0.8472	142.8	361.40	57.020	0.6902
220	98.8	288.64	57.048	0.8415	142.3	359.38	58.113	0.6916
230	97.9	287.91	54.414	0.8356	141.7	359.62	58.693	0.6846
240	97.2	287.56	55.569	0.8280	140.9	359.09	56.458	0.6807
250	96.0	284.10	56.606	0.8332	140.5	357.42	55.647	0.6815
260	95.4	284.01	57.098	0.8254	139.9	359.08	57.788	0.6692
270	94.6	283.57	56.834	0.8194	139.5	358.96	59.135	0.6641
280	93.6	281.38	56.271	0.8199	138.8	358.39	60.507	0.6608
290	93.1	280.20	55.649	0.8167	138.6	356.93	57.124	0.6617
300	92.4	280.18	55.073	0.8091	138.1	356.76	56.671	0.6576
310	92.1	277.45	53.682	0.8127	137.8	356.20	58.581	0.6553
320	91.4	275.77	56.567	0.8122	137.5	355.81	58.033	0.6521
330	90.9	275.90	53.939	0.8044	137.1	356.39	57.499	0.6453
340	90.7	273.34	54.882	0.8076	136.8	358.18	56.669	0.6338
350	90.3	272.87	53.173	0.8028	136.2	358.94	56.907	0.6265
360	90.2	270.80	54.468	0.8043	135.6	357.07	55.994	0.6299
370	89.6	270.26	53.820	0.7998	135.5	354.29	55.026	0.6370
380	89.3	268.46	54.053	0.8004	135.2	358.88	57.276	0.6149
390	88.9	266.78	53.972	0.8011	134.7	357.00	57.201	0.6188
400	88.6	266.53	52.715	0.7960	134.6	357.23	57.376	0.6140
410	88.3	266.17	51.475	0.7915	133.8	358.01	54.041	0.6071
420	88.0	266.12	53.688	0.7856	133.6	355.48	56.261	0.6138
430	87.9	263.38	52.691	0.7911	132.9	355.14	55.518	0.6114
440	87.6	263.78	52.932	0.7841	132.7	354.86	54.583	0.6090
450	87.1	262.20	54.649	0.7853	132.3	355.09	57.650	0.6048
460	87.1	260.72	52.870	0.7859	131.8	356.49	54.744	0.5959
470	86.7	259.86	54.717	0.7844	131.5	356.80	56.119	0.5915
480	86.5	256.55	52.474	0.7929	131.3	356.66	54.646	0.5889
490	86.3	257.43	54.582	0.7843	131.0	355.58	54.873	0.5902
500	86.2	256.67	53.855	0.7823	130.8	358.01	57.125	0.5775

Table A.12: Comparison of Algorithm Performance for 100 Iterations, with and without Local Repair, of a Contiguous Districting Problem for the State of Alabama with a District Compactness Heuristic and Neighborhood Mutation with rate $p_m = 0.005$, and population size $\mu = 100$.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total Population Distribution and District Compactness, Part 1 of 2							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	1743.5	4448.25	676.83	0.9946	2013.3	4251.01	624.38	0.9853
20	1324.7	3876.65	676.79	0.9937	1622.1	3551.51	639.47	0.9822
30	1246.1	3472.27	706.68	0.9926	1126.2	2981.66	663.39	0.9782
40	1130.3	3108.60	703.04	0.9925	618.0	2511.93	715.78	0.9792
50	972.5	2805.01	729.89	0.9923	490.5	2098.48	709.39	0.9741
60	786.9	2541.58	695.28	0.9919	418.3	1752.83	640.22	0.9712
70	767.1	2311.33	689.43	0.9881	399.2	1456.20	553.18	0.9698
80	617.1	2027.23	684.31	0.9911	374.3	1227.59	477.02	0.9687
90	559.9	1820.47	661.82	0.9891	339.0	1054.71	420.13	0.9651
100	463.2	1639.81	661.54	0.9880	336.6	924.42	374.56	0.9643
110	361.9	1456.62	612.88	0.9865	315.7	840.75	327.80	0.9604
120	348.0	1309.21	559.41	0.9874	304.1	776.61	267.35	0.9626
130	340.0	1186.08	519.90	0.9864	291.6	734.60	226.19	0.9608
140	326.9	1072.39	481.99	0.9851	279.9	697.76	193.14	0.9612
150	323.6	964.97	438.13	0.9853	261.1	679.64	183.85	0.9605
160	295.2	873.82	411.60	0.9865	217.7	647.20	152.71	0.9604
170	295.2	791.36	354.27	0.9875	217.6	634.08	138.18	0.9611
180	275.1	740.45	329.51	0.9857	215.8	632.53	121.50	0.9577
190	245.0	707.25	294.53	0.9832	215.7	631.96	118.24	0.9560
200	240.6	659.09	257.58	0.9838	214.0	622.42	111.90	0.9552
210	222.9	626.75	234.42	0.9835	214.0	611.93	108.18	0.9580
220	222.9	602.18	210.73	0.9836	214.0	611.17	112.24	0.9553
230	221.3	586.15	196.73	0.9817	214.0	612.32	95.12	0.9580
240	217.1	562.21	185.48	0.9850	214.0	604.71	92.56	0.9573
250	204.9	552.37	169.54	0.9824	214.0	606.27	99.09	0.9563
260	204.4	552.68	174.48	0.9789	206.3	601.17	96.37	0.9585
270	204.4	540.49	149.49	0.9799	206.3	598.44	97.34	0.9570
280	203.5	524.12	137.94	0.9820	206.3	590.95	85.04	0.9603
290	202.2	523.61	129.76	0.9812	206.3	592.21	96.00	0.9569
300	202.2	516.26	127.87	0.9808	205.6	597.99	93.18	0.9575
310	202.2	500.80	124.03	0.9829	205.6	602.87	82.26	0.9534
320	202.2	505.02	118.64	0.9811	205.6	597.13	94.97	0.9560
330	202.2	494.92	114.69	0.9839	205.6	601.13	88.79	0.9547
340	198.9	499.65	110.62	0.9803	205.6	611.63	98.74	0.9492
350	198.9	487.19	115.28	0.9835	205.6	602.51	97.65	0.9537
360	198.9	486.21	108.68	0.9828	205.6	596.64	94.29	0.9542
370	198.9	486.29	105.11	0.9807	205.6	593.28	89.64	0.9581
380	198.9	481.38	104.97	0.9830	205.6	598.78	95.56	0.9567
390	198.9	483.94	102.99	0.9819	205.6	600.40	82.01	0.9551
400	198.9	484.84	103.08	0.9816	205.6	594.00	93.89	0.9551
410	198.9	477.78	97.12	0.9804	205.6	588.70	87.76	0.9573
420	198.9	476.41	96.19	0.9813	205.6	590.92	91.11	0.9557
430	198.9	479.86	102.60	0.9815	205.6	598.11	89.59	0.9537
440	198.9	477.87	99.32	0.9819	205.6	594.00	93.18	0.9525
450	198.9	470.47	92.60	0.9836	205.6	598.67	94.49	0.9536
460	198.9	476.38	99.58	0.9801	205.6	592.13	89.02	0.9552
470	198.9	473.83	99.14	0.9805	205.6	588.01	95.45	0.9547
480	198.9	467.89	92.38	0.9817	205.6	601.85	89.43	0.9526
490	198.9	471.89	95.12	0.9827	205.6	591.78	91.64	0.9531
500	198.9	467.34	95.13	0.9826	205.6	593.14	91.51	0.9550

Table A.13: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total Population Distribution and District Compactness, Part 2 of 2							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	1770.1	4233.17	608.84	0.9610	1315.8	4350.02	630.09	0.9270
20	1415.9	3262.65	620.10	0.9552	751.3	3209.20	708.62	0.8997
30	808.1	2585.29	616.03	0.9462	413.2	2429.00	662.40	0.8855
40	486.7	2004.10	592.14	0.9381	395.6	1965.60	477.80	0.8567
50	461.4	1582.55	502.06	0.9326	408.4	1697.92	372.49	0.8450
60	323.1	1265.64	415.93	0.9314	366.7	1556.43	279.64	0.8278
70	307.3	1093.40	331.34	0.9223	346.7	1507.64	244.75	0.8177
80	293.4	979.61	268.83	0.9204	328.1	1523.36	230.44	0.8117
90	292.2	927.93	230.56	0.9176	283.3	1513.87	217.97	0.8024
100	270.2	894.32	201.94	0.9161	306.6	1508.32	253.64	0.7951
110	255.9	884.01	189.74	0.9083	326.6	1537.19	255.21	0.7864
120	250.9	849.73	165.26	0.9116	311.1	1545.97	244.48	0.7772
130	250.3	835.06	138.64	0.9139	321.4	1572.69	234.05	0.7753
140	234.2	824.62	131.58	0.9084	334.7	1592.52	265.56	0.7647
150	233.8	924.22	131.25	0.9095	303.6	1585.55	261.80	0.7676
160	233.1	833.11	133.90	0.9061	292.3	1583.94	274.01	0.7725
170	233.1	819.53	131.03	0.9079	318.4	1591.20	266.65	0.7633
180	233.0	815.76	120.49	0.9074	334.8	1620.54	300.03	0.7593
190	231.9	819.68	137.67	0.9053	350.8	1649.39	296.90	0.7549
200	231.9	813.58	121.47	0.9069	344.0	1692.52	296.04	0.7378
210	231.9	817.71	126.23	0.9074	342.5	1691.79	283.75	0.7404
220	228.0	819.49	129.31	0.9077	291.8	1722.99	317.79	0.7384
230	227.2	816.63	131.04	0.9053	348.9	1714.64	278.52	0.7333
240	227.2	812.93	115.56	0.9070	357.7	1724.74	312.99	0.7311
250	227.2	813.19	134.42	0.9042	342.8	1719.08	293.14	0.7287
260	227.2	810.53	124.67	0.9057	319.2	1742.15	319.81	0.7251
270	227.2	810.02	128.69	0.9090	312.3	1767.45	327.59	0.7239
280	227.2	810.94	117.30	0.9040	314.2	1780.98	332.16	0.7205
290	227.2	819.40	119.99	0.9049	296.4	1768.66	312.61	0.7181
300	227.2	808.35	114.03	0.9033	316.8	1775.18	314.41	0.7142
310	227.2	815.03	118.14	0.8996	343.5	1796.20	323.33	0.7100
320	226.8	812.63	135.86	0.9006	327.2	1853.02	354.99	0.6933
330	226.8	808.53	113.44	0.9045	321.7	1870.58	362.43	0.6906
340	226.8	809.75	122.50	0.9040	363.1	1883.55	347.97	0.6935
350	226.9	821.10	123.45	0.8993	311.9	1879.47	371.47	0.6899
360	226.8	811.26	118.20	0.9043	306.7	1891.65	374.46	0.6855
370	226.8	818.75	115.22	0.9055	376.3	1900.47	382.33	0.6856
380	226.8	818.48	127.50	0.9024	361.1	1934.31	377.41	0.6755
390	226.8	814.27	137.95	0.9051	339.4	1916.47	381.43	0.6817
400	226.8	807.56	134.63	0.9032	372.6	1933.84	362.70	0.6808
410	226.8	806.85	121.02	0.9018	331.8	1945.92	385.92	0.6785
420	226.8	823.16	130.83	0.9034	348.0	1970.81	438.71	0.6698
430	226.8	803.71	119.94	0.8979	367.6	1979.71	423.58	0.6622
440	226.8	808.11	109.40	0.9026	325.6	1992.16	409.55	0.6629
450	226.8	823.23	133.15	0.9006	296.7	2016.17	418.52	0.6573
460	226.8	822.93	131.46	0.9069	375.3	2017.86	426.79	0.6652
470	226.8	810.46	114.43	0.9019	304.1	2017.98	418.97	0.6563
480	226.8	817.46	129.11	0.9010	430.7	2027.08	443.23	0.6565
490	226.8	818.98	120.13	0.9015	335.9	2043.18	421.81	0.6467
500	226.8	820.10	119.79	0.8992	396.8	2056.74	408.20	0.6478

Table A.14: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Minority Population Distribution and District Compactness, Part 1 of 2							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	1986.7	3046.51	277.36	0.9946	2024.1	2988.35	251.29	0.9858
20	1565.4	2837.56	298.45	0.9933	1873.4	2701.24	266.37	0.9848
30	1485.9	2709.75	305.24	0.9945	1734.5	2508.91	272.51	0.9823
40	1377.8	2595.31	324.26	0.9922	1564.1	2360.36	277.18	0.9797
50	1367.1	2502.10	322.00	0.9924	1304.9	2242.69	296.62	0.9745
60	1365.6	2412.98	316.48	0.9924	1226.8	2107.18	307.04	0.9734
70	1284.2	2335.11	311.47	0.9918	1042.1	1998.58	314.21	0.9691
80	1260.7	2262.39	316.59	0.9891	959.8	1901.86	325.88	0.9676
90	1091.1	2178.23	329.78	0.9876	854.5	1804.80	331.02	0.9638
100	996.4	2107.64	328.32	0.9882	612.0	1703.16	347.33	0.9601
110	959.7	2034.55	333.45	0.9877	500.8	1604.62	360.41	0.9623
120	836.0	1977.47	340.15	0.9848	472.5	1520.31	364.67	0.9585
130	824.7	1911.33	333.05	0.9864	350.2	1439.62	362.97	0.9516
140	671.0	1845.46	332.96	0.9849	321.2	1366.89	364.87	0.9488
150	597.4	1787.41	331.92	0.9827	306.3	1293.10	361.07	0.9523
160	501.8	1726.55	337.86	0.9838	301.8	1229.19	354.79	0.9513
170	468.6	1663.21	352.14	0.9850	285.3	1174.12	346.32	0.9461
180	441.5	1614.41	361.54	0.9831	281.2	1127.44	343.28	0.9402
190	422.3	1565.25	367.59	0.9853	271.6	1077.79	344.69	0.9398
200	410.3	1514.00	365.48	0.9810	260.1	1027.74	338.40	0.9387
210	406.2	1460.77	371.10	0.9800	244.7	974.71	339.00	0.9404
220	400.9	1411.10	378.06	0.9809	242.0	935.24	334.52	0.9352
230	391.0	1364.97	381.92	0.9808	236.5	889.12	326.29	0.9362
240	391.0	1322.40	385.18	0.9796	227.7	854.10	314.75	0.9314
250	378.8	1279.02	381.61	0.9879	217.3	818.35	299.31	0.9290
260	374.3	1242.25	382.22	0.9779	214.4	783.18	292.89	0.9303
270	369.0	1203.43	386.70	0.9799	213.1	754.86	284.73	0.9273
280	364.4	1163.08	393.36	0.9780	212.1	730.43	278.41	0.9253
290	358.6	1129.75	393.40	0.9752	211.0	697.07	268.88	0.9281
300	345.9	1092.97	391.54	0.9752	209.4	678.88	260.92	0.9183
310	334.5	1059.00	393.41	0.9773	207.3	652.54	252.62	0.9261
320	324.5	1026.86	391.76	0.9773	207.0	632.90	244.84	0.9229
330	318.6	995.75	388.99	0.9756	204.8	611.16	236.69	0.9268
340	310.8	961.17	386.68	0.9772	204.8	601.54	226.50	0.9209
350	293.7	935.87	388.24	0.9711	204.8	586.94	216.01	0.9174
360	289.9	911.89	387.05	0.9736	204.8	564.13	201.80	0.9245
370	280.2	881.05	380.32	0.9732	204.8	550.28	193.03	0.9204
380	279.8	855.61	383.33	0.9730	204.8	533.19	180.97	0.9235
390	277.8	832.13	379.29	0.9742	204.8	524.38	175.74	0.9174
400	272.4	810.46	374.21	0.9719	204.8	516.15	166.87	0.9176
410	271.5	784.03	371.49	0.9766	204.8	504.19	155.83	0.9185
420	270.0	764.05	370.53	0.9738	204.8	496.61	151.99	0.9210
430	269.6	747.45	370.98	0.9723	204.8	492.55	146.86	0.9151
440	267.3	730.15	369.16	0.9705	204.8	485.03	143.40	0.9132
450	263.3	711.97	364.40	0.9722	204.8	478.82	136.66	0.9157
460	255.9	696.32	360.97	0.9757	204.8	470.98	132.80	0.9171
470	251.5	684.27	361.86	0.9715	204.8	467.62	123.93	0.9108
480	245.3	671.52	359.00	0.9735	204.8	463.64	122.63	0.9123
490	244.5	658.31	353.81	0.9685	204.8	461.59	122.30	0.9103
500	241.7	644.74	345.32	0.9704	204.8	455.91	119.32	0.9150

Table A.15: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of minority population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Minority Population Distribution and District Compactness, Part 2 of 2							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	2004.3	2890.86	282.65	0.9675	1954.7	2909.15	265.63	0.9314
20	1323.9	2572.75	302.02	0.9619	1370.8	2543.89	307.73	0.9069
30	1141.4	2365.28	302.01	0.9545	1146.1	2309.46	322.15	0.8803
40	1049.0	2189.60	325.25	0.9453	1009.4	2103.45	337.14	0.8564
50	893.5	2044.41	341.45	0.9307	808.6	1936.21	337.68	0.8401
60	770.6	1900.30	350.34	0.9252	599.2	1787.63	363.80	0.8088
70	551.5	1759.76	343.21	0.9252	552.3	1638.15	378.25	0.7939
80	525.9	1635.12	333.36	0.9174	436.8	1515.09	375.94	0.7773
90	510.0	1518.20	330.14	0.9126	397.3	1409.64	327.00	0.7688
100	494.1	1412.42	328.09	0.9010	370.9	1324.74	377.27	0.7415
110	491.8	1320.26	336.50	0.8935	335.9	1236.94	361.89	0.7270
120	470.5	1229.02	325.24	0.8904	298.5	1167.03	358.17	0.7101
130	442.7	1155.35	324.09	0.8830	270.8	1113.88	345.62	0.6907
140	343.4	1077.73	319.66	0.8847	268.7	1052.41	328.49	0.6949
150	306.2	1022.35	333.21	0.8664	257.2	1017.64	320.90	0.6768
160	295.8	968.12	327.77	0.8646	244.3	984.05	310.11	0.6613
170	298.2	911.97	314.95	0.8644	250.0	950.21	276.53	0.6554
180	290.8	867.52	308.39	0.8596	261.2	925.78	257.38	0.6398
190	282.5	824.78	300.70	0.8535	265.3	901.29	242.02	0.6361
200	274.7	790.30	290.88	0.8538	273.5	887.65	230.79	0.6299
210	266.7	747.13	262.14	0.8440	255.4	880.17	216.23	0.6233
220	261.4	710.74	238.29	0.8441	252.3	866.96	208.73	0.6211
230	255.7	684.85	223.60	0.8367	254.8	846.67	196.39	0.6096
240	253.9	662.28	217.38	0.8375	251.5	843.03	174.59	0.5965
250	248.9	648.20	207.72	0.8334	245.0	835.60	168.80	0.6004
260	224.0	622.79	189.30	0.8426	245.3	835.08	156.69	0.5905
270	211.8	616.81	188.40	0.8235	259.1	828.89	144.14	0.5847
280	210.4	604.22	173.56	0.8235	251.7	830.98	142.74	0.5733
290	208.7	585.12	174.36	0.8267	257.0	824.33	133.82	0.5738
300	208.7	583.21	149.19	0.8226	242.7	832.36	138.79	0.5619
310	208.7	570.32	131.54	0.8229	251.0	824.17	129.46	0.5733
320	208.0	556.94	125.55	0.8210	243.7	821.11	129.73	0.5624
330	206.8	554.15	123.33	0.8246	233.3	816.47	115.17	0.5575
340	204.5	554.44	115.78	0.8189	233.3	830.12	124.64	0.5537
350	204.5	551.49	110.00	0.8163	235.2	829.27	130.25	0.5431
360	204.5	537.99	114.09	0.8213	234.5	821.68	127.67	0.5391
370	204.1	541.14	103.84	0.8114	246.1	835.00	117.93	0.5301
380	204.1	536.59	100.01	0.8178	231.2	830.83	124.30	0.5415
390	204.1	529.14	98.08	0.8189	233.7	843.96	117.14	0.5232
400	204.1	534.48	99.89	0.8108	235.8	833.78	122.11	0.5408
410	204.1	530.80	98.65	0.8202	237.3	838.17	122.86	0.5196
420	204.1	528.75	97.18	0.8195	256.8	841.24	121.82	0.5145
430	204.1	532.04	91.43	0.8145	239.9	833.81	122.58	0.5207
440	204.1	525.46	90.68	0.8128	248.8	849.55	122.43	0.5086
450	204.1	526.63	88.99	0.8148	236.0	844.42	119.61	0.5200
460	204.1	526.81	88.60	0.8122	238.7	846.21	123.51	0.5054
470	204.1	523.59	90.07	0.8191	237.2	843.04	123.02	0.5030
480	204.1	532.00	81.60	0.8122	251.1	856.52	117.82	0.4903
490	204.1	526.44	83.69	0.8088	258.2	843.48	124.50	0.5049
500	204.1	527.54	79.36	0.8109	254.0	855.36	136.90	0.4860

Table A.16: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of minority population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total and Minority Population Distribution, Part 1 of 2							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	49579.4	68703.13	6431.85	0.9939	49545.6	65570.86	5904.88	0.9859
20	45130.4	63649.63	5922.94	0.9929	45962.5	59897.78	5516.54	0.9817
30	41951.5	60091.79	5697.61	0.9926	42676.9	56686.73	5177.94	0.9793
40	41535.6	57965.98	5595.80	0.9922	42176.4	55018.89	5018.50	0.9775
50	41535.6	56417.48	5601.08	0.9905	40914	53461.91	5000.94	0.9777
60	41509.6	55286.49	5644.01	0.9910	39378.1	52347.84	5087.08	0.9735
70	40135.5	54404.56	5656.31	0.9892	37607.6	51427.12	5246.83	0.9679
80	38847.6	53690.49	6029.86	0.9886	36925.3	50575.85	5437.05	0.9677
90	37025.3	53158.38	6260.62	0.9890	33429.6	49963.73	5729.74	0.9655
100	34731.1	52700.29	6393.14	0.9886	33140.8	49281.72	5792.47	0.9657
110	33580	52262.82	6461.07	0.9886	31851.3	48810.06	5990.18	0.9594
120	33278.4	51855.54	6510.87	0.9875	31544.2	48113.21	6110.81	0.9642
130	32693.5	51509.78	6570.66	0.9866	31319.8	47682.69	6196.57	0.9630
140	32479.2	51217.84	6531.68	0.9842	30008.7	47134.55	6250.38	0.9634
150	32475.6	50806.95	6545.58	0.9865	29221.5	46890.30	6414.58	0.9594
160	32056.8	50489.02	6472.78	0.9866	28827.9	46677.33	6350.16	0.9543
170	31499	50155.75	6584.06	0.9869	28052.5	46316.66	6517.86	0.9607
180	31041.3	49897.30	6610.43	0.9864	27990.7	46138.98	6634.10	0.9509
190	30048.3	49669.94	660.82	0.9853	27656.5	46023.46	662.15	0.9514
200	29441.2	49428.31	6827.36	0.9842	27170.5	45713.86	6694.49	0.9530
210	28141.2	49117.24	6891.65	0.9855	27067.2	45486.89	6846.24	0.9506
220	25653.6	48966.53	7052.88	0.9824	27060.2	45266.55	7146.11	0.9488
230	24701.3	48736.36	7201.89	0.9835	26943.6	44935.22	7288.41	0.9510
240	24578	48597.54	7293.57	0.9822	26873.6	44790.65	7344.15	0.9469
250	24351.7	48450.79	7398.52	0.9825	25969.6	44619.70	7513.64	0.9457
260	24127.1	48237.22	7416.89	0.9837	25649.8	44472.72	7555.66	0.9440
270	23621.7	48096.61	7462.39	0.9828	25349.4	44281.97	7454.84	0.9444
280	23611.6	48006.75	7495.96	0.9820	25221.8	44069.35	7489.90	0.9395
290	23587.1	47818.24	7514.59	0.9816	25172.3	43840.28	7676.78	0.9448
300	23535.6	47703.30	7572.10	0.9822	24771.7	43671.45	7748.32	0.9434
310	23339.6	47647.25	7642.56	0.9806	24747.3	43616.59	7642.33	0.9415
320	23339.6	47480.30	7563.26	0.9808	24747.3	43431.88	7710.86	0.9410
330	23199.4	47252.90	7666.68	0.9833	24516	43243.05	7851.97	0.9402
340	23199.4	47199.80	7734.62	0.9825	24510.9	43006.74	7793.86	0.9432
350	23189.9	47158.05	7740.08	0.9809	24294.1	42962.82	7897.92	0.9375
360	23062.3	470303.21	7760.72	0.9814	23423.2	42761.87	7922.01	0.9405
370	23062.3	46994.15	7720.97	0.9801	23100.1	42661.63	7950.99	0.9368
380	23049.9	46903.58	7756.42	0.9807	21829.4	42551.35	8050.29	0.9337
390	23037.4	46866.86	7803.06	0.9797	21189.4	42483.45	8320.96	0.9311
400	22958.6	46759.29	7896.87	0.9813	20527	42220.35	8251.75	0.9364
410	22716.5	46720.49	7933.60	0.9800	20438.1	42156.18	8303.14	0.9320
420	21680.4	46548.97	8018.06	0.9824	20102.5	41891.68	8369.46	0.9373
430	21518.2	46519.23	8030.45	0.9806	19760.1	41829.75	8356.42	0.9312
440	20699.1	46364.55	8126.43	0.9828	16950.1	41793.60	8416.42	0.9315
450	20355.1	46266.01	8096.23	0.9792	16361.2	41617.28	8636.49	0.9299
460	20282	46202.30	8132.72	0.9795	15904	41551.09	8609.94	0.9245
470	20282	46158.32	8170.26	0.9774	15007.3	41450.55	8779.12	0.9252
480	20269.5	46120.66	8206.60	0.9766	14815.8	41299.40	8762.04	0.9267
490	20269.5	46001.61	8129.17	0.9776	14809.1	41170.92	8741.29	0.9265
500	20261.5	45910.19	8157.03	0.9783	14809.1	40952.61	8915.32	0.9301

Table A.17: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population and minority population with a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total and Minority Population Distribution, Part 2 of 2							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	41481.4	65371.65	5967.90	0.9689	44047.6	64179.04	5704.90	0.9301
20	40132.5	58293.25	5256.16	0.9630	35097.8	57373.85	5912.86	0.9090
30	38598.9	54929.09	5412.75	0.9520	31681.2	54036.18	6119.29	0.8908
40	35599.1	52773.24	5746.00	0.9462	30660.1	51843.92	5932.63	0.8655
50	34076.4	51195.53	5827.98	0.9416	28241.7	49968.02	5814.10	0.8504
60	33033.3	50108.11	5982.12	0.9315	23626.4	48796.78	5859.67	0.8300
70	32223.1	49128.14	5911.99	0.9277	22281.8	47461.21	6029.76	0.8196
80	31264.8	48089.37	6308.86	0.9267	20132.5	46409.18	6111.01	0.7973
90	30507.8	47473.20	6450.94	0.9138	17854.8	45113.54	6243.24	0.7862
100	28974.6	46630.74	6453.91	0.9137	16453.1	44212.93	6170.80	0.7603
110	27879.8	45825.73	6487.94	0.9034	15332.3	43245.74	6225.65	0.7494
120	25703.7	44821.33	6734.94	0.9066	15177.7	42215.05	6475.64	0.7409
130	24962.9	44284.19	6841.74	0.9007	12759.5	41624.84	6375.51	0.7294
140	24669.9	43616.80	6975.09	0.8987	13078.5	40758.64	6528.13	0.7161
150	24287.7	43199.21	7334.80	0.8909	13106.1	40048.89	6616.28	0.6989
160	23827.7	42687.42	7369.94	0.8880	12810.3	39168.10	6564.36	0.6964
170	23078.2	42177.43	7595.82	0.8833	12471.5	38749.82	6777.58	0.6766
180	22583.4	41668.61	7735.00	0.8801	12495.5	37998.78	6679.16	0.6731
190	22513.3	41442.35	7831.47	0.8729	12588.9	37635.27	6732.65	0.6557
200	22346.1	40883.13	7994.25	0.8749	12419.4	36773.40	6475.06	0.6564
210	21875.9	40593.06	8218.97	0.8663	12822.0	36546.30	6653.74	0.6367
220	21470.5	40150.91	8210.58	0.8619	12424.3	35916.08	6641.27	0.6305
230	20236.9	39864.33	8333.24	0.8570	13148.8	35778.75	6770.64	0.6129
240	19935.5	39332.59	8387.10	0.8584	13156.0	35138.04	6741.89	0.6180
250	19783.8	39029.93	8484.85	0.8549	13923.8	35172.89	7000.11	0.5949
260	18196.3	38638.72	8533.29	0.8545	13244.7	34883.96	6571.64	0.5778
270	17790.4	38475.58	8705.06	0.8493	12837.3	34262.52	6759.39	0.5871
280	17586.7	37949.33	8724.08	0.8482	12516.1	33992.01	6656.01	0.5746
290	16543.5	37412.62	8690.43	0.8510	12203.3	33986.10	6701.01	0.5695
300	15939.5	37019.01	8641.86	0.8504	12450.7	33472.11	6619.05	0.5636
310	15932.2	36978.42	8771.22	0.8395	12154.6	33527.35	6533.46	0.5515
320	15804.3	36440.47	8831.08	0.8396	11589.5	33365.79	6449.66	0.5450
330	15658.1	36299.63	8817.67	0.8306	11937.2	33044.41	6538.26	0.5259
340	14744.0	35923.64	8947.63	0.8304	11665.3	32885.53	6396.00	0.5317
350	14394.3	35691.68	9056.02	0.8243	12212.1	32875.88	6722.85	0.5168
360	14201.8	35516.55	9112.90	0.8193	12321.5	32528.73	6508.63	0.5105
370	13997.8	35161.65	9089.08	0.8215	12422.4	32444.07	6755.09	0.5127
380	13997.8	35060.38	8902.32	0.8172	11376.0	32228.15	6507.39	0.5043
390	13997.8	34883.25	9070.78	0.8128	11245.7	32126.75	6269.99	0.4923
400	13955.4	34660.43	8905.95	0.8034	11671.8	32057.15	6409.84	0.4894
410	13613.7	34388.10	9133.03	0.8064	12496.0	62206.00	6183.56	0.4797
420	13613.7	34196.66	9077.24	0.8069	12455.9	31910.55	6461.44	0.4790
430	13515.2	34076.51	9285.76	0.8015	11937.7	31805.03	6221.55	0.4756
440	13254.1	33732.06	9251.19	0.8038	11763.5	31868.69	6259.31	0.4679
450	12992.5	33653.64	9221.87	0.7981	11942.9	31707.77	6192.62	0.4691
460	12569.7	33384.55	9184.36	0.8008	10304.3	31329.58	6058.31	0.4641
470	12415.6	33049.78	9063.09	0.8055	11345.6	31297.28	5876.91	0.4583
480	12294.1	32970.45	9366.70	0.7927	11177.3	31219.40	5467.41	0.4626
490	12056.7	32624.13	9204.79	0.7921	10398.2	31152.39	5600.04	0.4637
500	11701.9	32531.59	9351.44	0.7898	10070.9	31115.81	5525.17	0.4544

Table A.18: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population and minority population with a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total and Minority Population Distribution and District Compactness, Part 1 of 2							
	Mutation Rate $p_m = 0.001$				Mutation Rate $p_m = 0.0025$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	52737.9	67470.10	5668.49	0.9943	50917.4	67283.71	5209.93	0.9857
20	47912.0	62725.42	5459.73	0.9923	45806.3	60890.04	4815.34	0.9816
30	46772.6	60068.37	5429.89	0.9926	39888.9	57284.42	5059.42	0.9789
40	43563.5	58061.76	5643.71	0.9921	38858.4	55304.53	5138.30	0.9787
50	41759.8	56679.05	5849.66	0.9919	38410.4	53818.38	5316.53	0.9763
60	40602.5	55730.85	5853.97	0.9905	38140.5	52748.72	5485.03	0.9731
70	38969.0	55069.65	6004.10	0.9893	37855.3	51789.60	5688.68	0.9737
80	38573.9	54318.74	6182.47	0.9902	35034.8	51177.06	5781.63	0.9677
90	38538.3	53792.73	6159.82	0.9880	34264.8	50382.51	6084.71	0.9686
100	38402.6	53323.10	6155.49	0.9892	33459.4	49797.39	6220.14	0.9644
110	37884.6	52967.24	6133.09	0.9887	33183.8	49146.69	6290.66	0.9646
120	37789.2	52654.99	6152.21	0.9873	32623.3	48579.37	6445.02	0.9640
130	37636.0	52320.22	6078.20	0.9875	31377.3	48196.71	6616.55	0.9602
140	37296.8	52062.01	6083.54	0.9878	30422.0	47740.92	6533.96	0.9617
150	36865.9	51870.96	6061.18	0.9874	29932.6	47464.20	6705.37	0.9575
160	36865.9	51699.41	6097.44	0.9866	28860.4	47168.22	6604.92	0.9569
170	36865.9	51569.43	6085.62	0.9864	28018.4	46901.37	6668.34	0.9529
180	36673.1	51460.57	6021.43	0.9829	27106.4	46646.20	6818.00	0.9524
190	35898.2	51217.77	5980.81	0.9856	26319.3	46484.67	6825.32	0.9505
200	35796.5	51198.17	6093.88	0.9821	25896.3	46189.28	7053.07	0.9512
210	35770.5	50961.35	5989.00	0.9852	25229.3	45929.67	7017.85	0.9503
220	35770.5	50937.54	6069.63	0.9828	23708.7	45692.11	7073.07	0.9509
230	35760.8	50730.46	5993.43	0.9852	23077.5	45538.29	7184.38	0.9446
240	35555.2	50668.40	6017.20	0.9839	23054.5	45241.77	7292.48	0.9467
250	34396.2	50567.27	6039.08	0.9852	22525.2	45203.59	7295.90	0.9414
260	33668.9	50456.13	6069.39	0.9834	21202.8	44878.94	7380.64	0.9420
270	33633.4	50341.59	6020.35	0.9815	20505.3	44662.13	7548.28	0.9445
280	33625.0	50116.53	6079.23	0.9825	20380.9	44462.49	7517.93	0.9395
290	33625.0	50009.73	6090.66	0.9832	20302.2	44179.19	7545.97	0.9442
300	33625.0	49957.43	6165.97	0.9812	20019.0	44158.71	7891.22	0.9368
310	33625.0	49823.90	6153.48	0.9821	18210.9	44045.00	7817.34	0.9332
320	33625.0	49723.36	6222.25	0.9818	18176.1	43731.74	7861.11	0.9390
330	32665.6	49593.32	6270.78	0.9826	16461.2	43642.32	7840.40	0.9370
340	32393.1	49599.45	6246.71	0.9790	15899.8	43453.28	8089.09	0.9376
350	32232.1	49428.67	6313.80	0.9826	15892.2	43306.90	8027.19	0.9331
360	32220.4	49356.26	6364.78	0.9836	15844.4	43099.86	8226.96	0.9342
370	32148.0	49428.89	6428.55	0.9790	15747.1	43136.27	8192.38	0.9262
380	32145.8	49335.91	6416.15	0.9804	15745.6	43021.07	8303.94	0.9285
390	32143.8	49241.32	6424.64	0.9827	15744.0	42923.12	8266.40	0.9266
400	31956.1	49131.94	6469.36	0.9798	15743.4	42723.99	8318.43	0.9280
410	31432.7	48899.01	6551.59	0.9782	15580.6	42523.75	8380.35	0.9325
420	31424.9	48722.14	6616.10	0.9791	15577.8	42528.84	8437.54	0.9239
430	31421.2	48629.05	6664.15	0.9796	15577.0	42296.84	8534.31	0.9264
440	31391.0	48547.58	6645.24	0.9804	15575.8	42000.88	8488.08	0.9307
450	30836.6	48413.31	6785.72	0.9808	15575.3	41928.53	8524.85	0.9257
460	30303.8	48391.35	6803.70	0.9792	15574.2	41876.06	8523.97	0.9216
470	29497.8	48264.72	6773.39	0.9811	15573.5	41716.60	8751.53	0.9225
480	27539.5	48209.60	6918.34	0.9791	15572.2	41567.35	8670.17	0.9204
490	26452.7	48138.27	7022.40	0.9799	15571.7	41458.60	8763.14	0.9215
500	24562.2	48046.61	7175.40	0.9779	15571.0	41362.42	8745.61	0.9225

Table A.19: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population and minority population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with a Multi-Objective Heuristic Measure of Total and Minority Population Distribution and District Compactness, Part 2 of 2							
	Mutation Rate $p_m = 0.005$				Mutation Rate $p_m = 0.010$			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	46878.9	65658.55	5838.04	0.9689	45152.4	65405.04	5277.57	0.9281
20	40685.6	59433.00	5194.25	0.9628	39946.7	58085.38	5003.77	0.9055
30	39559.1	56017.78	4982.96	0.9494	36584.4	54695.77	5091.82	0.8871
40	38522.3	53819.65	5411.83	0.9416	34862.7	52740.33	5032.44	0.8646
50	32798.8	52053.49	5515.30	0.9441	31246.5	51161.01	5481.13	0.8413
60	31177.8	51024.95	5923.76	0.9309	28180.0	49943.86	5718.20	0.8236
70	30034.2	49930.94	5778.00	0.9310	25077.2	48184.51	6214.13	0.8131
80	28556.4	49042.89	5840.96	0.9229	23684.8	47083.62	6681.64	0.7914
90	27268.9	48345.01	5916.13	0.9155	23207.9	45930.82	7032.57	0.7806
100	26562.3	47584.36	5975.34	0.9153	22710.4	45131.35	6808.36	0.7660
110	26063.1	46963.38	5976.02	0.9092	22042.9	44209.77	6997.67	0.7532
120	25996.2	46372.76	6228.18	0.9108	21415.9	43513.61	6945.23	0.7317
130	25181.0	45726.70	6268.42	0.9040	16674.1	42914.58	7180.48	0.7191
140	24899.4	45193.14	6601.27	0.8961	16671.6	42142.15	7170.97	0.7033
150	24252.8	44592.99	6710.27	0.8940	16106.4	41328.04	7114.37	0.6888
160	23257.1	44010.07	6849.46	0.8897	15886.5	40865.81	7527.27	0.6767
170	22721.6	43478.54	6978.62	0.8880	15882.7	39969.37	7200.14	0.6756
180	22242.8	42993.77	7130.43	0.8792	15996.7	39885.57	7421.90	0.6525
190	20267.6	42550.13	7202.53	0.8753	15829.3	39027.69	7350.09	0.6493
200	17354.1	42188.83	7287.19	0.8679	15687.3	38761.54	7374.20	0.6252
210	16836.2	41421.63	7427.92	0.8668	15367.4	38141.33	7265.17	0.6374
220	16836.2	41008.66	7635.03	0.8626	15508.3	37761.65	7425.58	0.6237
230	16794.1	40650.98	7789.45	0.8598	14970.3	37304.71	7482.98	0.6103
240	16688.1	40411.86	7850.89	0.8501	14212.3	36756.42	7259.42	0.6063
250	15818.0	39928.15	7920.56	0.8470	13951.5	36388.66	7462.19	0.5894
260	15216.8	39398.26	7989.62	0.8446	14817.1	36193.76	7567.89	0.5779
270	14657.3	38985.36	8061.78	0.8396	14762.2	35841.01	7353.62	0.5691
280	14304.6	38632.52	8071.96	0.8370	14685.5	35368.69	7070.90	0.5691
290	14079.0	38374.75	8018.88	0.8325	13575.9	34876.32	6923.67	0.5667
300	13966.9	37823.72	8212.71	0.8389	13475.0	34647.20	6881.60	0.5528
310	13966.9	37745.01	8126.99	0.8226	13198.7	34319.63	6850.78	0.5462
320	13942.4	37384.86	8315.64	0.8249	12865.9	33998.76	6592.07	0.5394
330	13905.4	36898.17	8254.54	0.8265	12589.9	33576.92	6632.78	0.5390
340	13270.4	36527.27	8496.30	0.8140	13636.2	33494.21	6143.67	0.5369
350	13269.5	35977.52	8521.08	0.8185	13027.9	33531.17	6344.96	0.5291
360	13258.4	35580.99	8435.28	0.8159	12601.7	33239.70	6463.11	0.5161
370	13171.9	35332.79	8713.96	0.8108	12732.7	32894.17	6285.68	0.5222
380	13053.0	35137.25	8843.29	0.8072	13000.6	32987.45	6083.68	0.5074
390	12740.7	34829.05	8903.75	0.8044	12830.1	32778.93	6282.51	0.5104
400	12348.2	34471.79	8876.58	0.8025	12774.9	32451.85	5967.50	0.5185
410	12168.9	34449.18	9189.05	0.7930	12989.0	32240.14	5791.52	0.5061
420	12044.5	34091.00	9082.43	0.7955	12997.5	32468.09	5451.38	0.4932
430	11827.1	33771.13	9050.98	0.7934	13150.9	32512.78	5804.71	0.4936
440	11825.7	33477.92	9169.16	0.7952	13336.0	32477.83	5687.57	0.4803
450	11823.2	33141.23	9390.91	0.7963	12083.3	32410.19	5824.57	0.4749
460	11823.9	33018.36	9093.87	0.7909	13150.3	32336.50	5451.93	0.4780
470	11813.0	32722.65	9391.50	0.7964	13710.2	31943.63	5227.50	0.4873
480	11822.9	32691.13	9172.36	0.7859	13811.6	31976.28	5269.15	0.4715
490	11821.4	32638.23	9357.67	0.7800	13055.4	32050.69	4899.46	0.4627
500	11817.2	32368.15	9229.74	0.7836	13151.1	31843.55	5179.30	0.4712

Table A.20: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multi-objective heuristic that attempts to balance equal distribution of total population and minority population with district compactness and a soft contiguity constraint.

	Districting Problem for the State of Alabama with an Agent-based Heuristic Measure of Total and Minority Population Distribution							
	Total Population Distribution Heuristic				Minority Population Distribution Heuristic			
Generation	Best	Mean	StdDev	% Connected	Best	Mean	StdDev	% Connected
10	48069	193982.93	87883.65	0.9540	19438	80435.52	27432.22	0.9547
20	27479	165131.49	89014.70	0.9443	10805	68564.21	28290.12	0.9500
30	8046	145252.75	89444.47	0.9373	4893	61657.38	29072.49	0.9426
40	2446	123426.03	89129.65	0.9247	2588	56821.90	29027.91	0.9320
50	1811	120689.68	88720.36	0.9251	1996	53161.81	29015.82	0.9247
60	755	112152.21	88302.38	0.9097	595	50174.49	29475.54	0.9137
70	752	104093.58	87182.52	0.8986	313	47218.86	29410.27	0.9075
80	837	97109.97	85796.40	0.8910	288	44404.45	29340.49	0.9016
90	755	90419.21	84089.70	0.8903	283	42542.88	29352.70	0.8868
100	755	85242.25	82430.01	0.8790	241	40615.49	28970.18	0.8826
110	755	81128.55	81603.00	0.8703	205	38584.78	28783.57	0.8779
120	646	76692.16	80044.53	0.8650	187	36667.97	28513.11	0.8701
130	533	72799.78	78351.97	0.8667	152	34540.76	28252.19	0.8747
140	435	69603.12	76873.03	0.8576	129	32942.95	28139.38	0.8729
150	439	66506.04	75237.51	0.8454	102	31860.99	28101.46	0.8622
160	404	63943.17	73793.37	0.8455	85	31179.75	27749.07	0.8495
170	351	61847.09	71565.25	0.8338	68	30664.99	27588.61	0.8505
180	351	59125.48	69484.90	0.8347	47	28901.07	26979.12	0.8501
190	351	57169.42	67547.53	0.8280	45	28201.10	26707.33	0.8395
200	351	54520.21	65105.61	0.8320	47	27331.61	25912.00	0.8393
210	330	52626.88	64149.31	0.8288	43	26510.15	25131.62	0.8346
220	295	50723.38	61781.96	0.8292	33	25698.80	24904.57	0.8352
230	250	49162.52	60633.81	0.8207	27	25165.31	24771.69	0.8281
240	283	47318.84	58673.13	0.8224	27	24594.89	24136.75	0.8263
250	240	45530.69	56502.12	0.8203	27	24293.25	24115.02	0.8181
260	240	44147.34	55268.23	0.8217	27	23705.96	24062.62	0.8199
270	240	42810.34	53293.33	0.8154	27	23115.98	23566.94	0.8207
280	240	41821.12	51842.33	0.8082	27	22810.22	23073.88	0.8166
290	240	40767.54	50819.95	0.8105	27	22263.98	22741.20	0.8192
300	240	38818.54	48418.39	0.8144	27	22068.45	22784.80	0.8164
310	233	37960.43	46985.92	0.8097	27	21619.75	22186.88	0.8173
320	200	36322.91	45653.58	0.8138	27	21172.23	21999.44	0.8182
330	240	36182.31	44855.30	0.8042	27	21349.45	21888.34	0.8071
340	200	35034.75	44370.25	0.8099	27	21002.11	21180.91	0.8094
350	188	34178.56	43265.33	0.8138	27	20586.34	20624.55	0.8107
360	188	34974.01	42606.26	0.7891	27	20236.75	21174.65	0.8117
370	146	33872.35	41846.41	0.7965	25	20300.91	20553.05	0.8028
380	146	33016.84	40701.80	0.7940	25	19556.13	19872.52	0.8151
390	180	32201.52	39261.64	0.7970	25	19472.82	19842.70	0.8099
400	146	31731.94	38129.51	0.7915	13	19516.90	18942.87	0.8059
410	140	31327.98	37719.85	0.7890	25	19219.63	19047.13	0.8066
420	140	30465.92	36445.60	0.7882	25	18975.42	19026.21	0.8059
430	140	29645.77	35080.03	0.7931	25	18830.82	18878.69	0.8039
440	140	29574.35	33433.45	0.7842	25	18338.26	18168.27	0.8092
450	140	28684.20	33382.91	0.7897	16	17976.69	18341.11	0.8104
460	140	28581.49	32195.14	0.7859	25	18437.65	17771.29	0.8000
470	140	27900.69	31729.34	0.7885	24	18102.54	17758.67	0.8019
480	140	27982.45	31625.64	0.7801	24	17681.76	17319.65	0.8063
490	140	27702.14	30372.86	0.7802	24	17686.67	16681.58	0.8029
500	140	27331.72	30268.45	0.7790	19	17485.41	16741.49	0.8048

Table A.21: Comparison of Algorithm Performance for two trials over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with an Agent-based heuristic that attempts to balance equal distribution of total population, left, and equal distribution of minority population, right, with district compactness and a soft contiguity constraint.

Districting Problem for the State of Alabama with Multiple Pools of Agents Assessing Total and Minority Population Distribution, District Compactness, and District Contiguity				
Generation	Best	Mean	StdDev	% Connected
10	102349.5	210269.86	69772.08	0.9664
20	95354.0	184776.24	63401.48	0.9700
30	97374.5	169830.42	64204.60	0.9536
40	97195.5	158372.58	64379.14	0.9291
50	94152.5	147834.66	60981.68	0.9245
60	88221.5	140530.54	59936.84	0.9236
70	83721.5	135884.31	59327.11	0.8864
80	74926.5	129885.98	55006.40	0.9164
90	73092.0	131124.40	55850.32	0.8309
100	66041.5	125918.56	54988.60	0.8500
110	66041.5	121708.74	53562.09	0.8991
120	63526.5	118499.87	53356.85	0.8927
130	63191.5	116422.98	51182.70	0.9036
140	61472.5	114231.03	53231.26	0.8791
150	60164.5	112933.33	50845.60	0.8718
160	59670.0	111524.18	48939.12	0.8655
170	58988.0	110847.92	50001.59	0.8645
180	59011.0	109714.26	49526.16	0.8627
190	58441.0	107870.12	47514.06	0.8855
200	57646.0	107099.10	46437.98	0.8682
210	57251.0	107698.25	44993.06	0.8355
220	57229.5	106108.91	45260.95	0.8445
230	56882.5	104873.27	43728.93	0.8427
240	53885.5	107733.57	47372.43	0.7682
250	51784.5	106751.20	47752.31	0.7736
260	53781.0	104828.22	48007.52	0.7855
270	55529.5	105671.13	46494.48	0.7509
280	54220.5	105331.80	46501.27	0.7445
290	52042.0	103016.60	47702.07	0.7755
300	50936.5	102292.65	46370.23	0.7809
310	50100.0	102714.31	46072.60	0.7591
320	49388.5	100759.93	46576.95	0.7791
330	49248.5	100886.80	45885.67	0.7673
340	48845.5	100234.31	46004.86	0.7727
350	48698.5	101085.46	45717.88	0.7527
360	48658.5	102065.45	46480.09	0.7391
370	48377.0	100073.66	46004.44	0.7727
380	48630.5	101292.57	45423.59	0.7364
390	48443.0	98286.49	45564.51	0.7709
400	48443.0	102889.28	48768.41	0.6918
410	48443.0	103747.40	47962.88	0.6745
420	48443.5	101990.77	48799.44	0.7055
430	48437.5	101675.93	49375.87	0.7091
440	48409.5	103296.33	48669.33	0.6773
450	47911.5	103517.32	49156.33	0.6809
460	47682.0	103340.53	49332.15	0.6836
470	47619.5	104820.47	50486.66	0.6818
480	47520.5	104213.57	49552.36	0.6782
490	47385.0	104471.84	50434.91	0.6718
500	46324.0	105938.24	51069.29	0.6609

Table A.22: Comparison of Algorithm Performance over 100 Iterations of a Contiguous Districting Problem for the State of Alabama with a multiple pools of intelligent agents that attempt to balance equal distribution of total population, equal distribution of minority population, district compactness, and district contiguity.