SPATIAL OPTIMIZATION METHODS AND SYSTEM FOR REDISTRICTING PROBLEMS

by

Hai Jin

Bachelor of Engineering
Wuhan University, 2006

Master of Arts
Wuhan University, 2008

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Geography

College of Arts and Sciences

University of South Carolina

2017

Accepted by:

Diansheng Guo, Major Professor

Michael Hodgson, Committee Member

Cuizhen (Susan) Wang, Committee Member

Joshua Cooper, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

ACKNOWLEDGEMENTS

I'm sincerely grateful to all the people who have supported me during this journey.

First and foremost, I would like to thank my advisor, Dr. Diansheng Guo, for his continuous guidance, encouragement, and support throughout my doctoral work.

I wish to thank my committee members, Dr. Michael Hodgson, Dr. Cuizhen (Susan) Wang, and Dr. Joshua Cooper for their support and suggestions for my dissertation.

I wish to thank the department staff members and my fellow graduate students for their support and help.

Finally, I want to thank my parents for their unconditional support.

ABSTRACT

Redistricting is the process of dividing space into districts or zones while optimizing a set of spatial criteria under certain constraints. Example applications of redistricting include political redistricting, school redistricting, business service planning, and city management, among many others. Redistricting is a mission-critical component in operating governments and businesses alike. In research fields, redistricting (or region building) are also widely used, such as climate zoning, traffic zone analysis, and complex network analysis. However, as a combinatorial optimization problem, redistricting optimization remains one of the most difficult research challenges. There are currently few automated redistricting methods that have the optimization capability to produce solutions that meet practical needs. The absence of effective and efficient computational approaches for redistricting makes it extremely time-consuming and difficult for an individual person to consider multiple criteria/constraints and manually create solutions using a trial-and-error approach.

To address both the scientific and practical challenges in solving real-world redistricting problems, this research advances the methodology and application of redistricting by developing a new computational spatial optimization method and a system platform that can address a wide range of redistricting problems, in an automated and computation-assisted manner. The research has three main contributions. First, an efficient and effective spatial optimization method is developed for redistricting. The new

method is based on a spatially constrained and Tabu-based heuristics, which can optimize multiple criteria under multiple constraints to construct high-quality optimization solutions. The new approach is evaluated with real-world redistricting applications and compared with existing methods. Evaluation results show that the new optimization algorithm is more efficient (being able to allow real-time user interaction), more flexible (considering multiple user-expressed criteria and constraints), and more powerful (in terms of optimization quality) than existing methods. As such, it has the potential to enable general users to perform complex redistricting tasks.

Second, a redistricting system, ***iRedistrict***, is developed based on the newly developed spatial optimization method to provide user-friendly visual interface for defining redistricting problems, incorporating domain knowledge, configuring optimization criteria and methodology parameters, and ultimately meeting the needs of real-world applications for tackling complex redistricting tasks. It is particularly useful for users of different skill levels, including researchers, practitioners, and the general public, and thus enables public participation in challenging redistricting tasks that are of immense public interest. Performance evaluations with real-world case studies are carried out. Further computational strategies are developed and implemented to handle large datasets.

Third, the newly developed spatial optimization method is extended to solve a different spatial optimization problem, i.e., spatial community structure detection in complex networks, which is to partition networks to discover spatial communities by optimizing an objective function. Moreover, a series of new evaluations are carried out with synthetic datasets. This set of evaluations is different from the previous evaluations

with case studies in that, the optimal solution is known with synthetic data and therefore it is possible to evaluate (1) whether the optimization method can discover the true pattern (global optima), and (2) how different data characteristics may affect the performance of the method. Evaluation results reveal that existing non-spatial methods are not robust in detecting spatial community structure, which may produce dramatically different outcomes for the same data with different characteristics, such as different spatial aggregations, sampling rates, or noise levels. The new optimization method with spatial constraints is significantly more stable and consistent. In addition to evaluations with synthetic datasets, a case study is also carried out to detect urban community structure with human movements, to demonstrate the application and effectiveness of the approach.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Geographic districting problems (a.k.a. redistricting, zoning, or regionalization problems in different application contexts) are to group small geographic units into larger districts to optimize an objective function (i.e., a set of criteria) under a set of constraints. From the perspective of optimization, they can be considered as combinatorial optimization problems, which are to find an optimal (or near-optimal) solution from a large set of alternatives (Papadimitriou and Steiglitz 1998). Different from other combinatorial optimization problems, geographic districting problems usually consider spatial criteria and constraints such as spatial contiguity and compactness, which are difficult to integrate with mathematical models commonly used in non-spatial combinatorial optimization methods such as integer programming. Redistricting optimization has been shown to be NP-hard (Puppe and Tasnadi 2008, Altman 1997).

Redistricting problems are encountered in many application domains such as political redistricting, school redistricting, and business service zone planning. The primary difference among these applications from the perspective of optimization is that the objective function and constraints being considered in the optimization process are different. For example, the criteria and constraints considered for political redistricting include geographic contiguity, equal population, majority-minority district, preserving communities of interest, and spatial compactness (Levitt and Foster 2008), while school

redistricting may consider other criteria such as travel distance for students, school capacity, and socioeconomic balance within and cross school districts.

Redistricting problems have attracted extensive research efforts in developing automated redistricting approaches based on clustering (Forrest 1964), location-allocation (Hess et al. 1965, Kalcsics, Nickel and Schröder 2005), space partitioning (Ricca, Scozzari and Simeone 2008, Novaes et al. 2009), integer programming (Caro et al. 2004), graph partitioning (Mehrotra, Johnson and Nemhauser 1998b), genetic algorithms (Forman 2002), Tabu search (Bozkaya, Erkut and Laporte 2003, Ricca and Simeone 2008), and simulate annealing (Browdy 1990, D'Amico et al. 2002). Since geographic districting is an NP-complete problem, no method can guarantee to find the best solution unless the problem is very small, for which an exhaustive search is possible.

Existing automated methods, however, mostly remain in the academic domain since they do not meet practical needs—the methods are either limited to small data sets or cannot produce results with sufficient optimization quality. Current redistricting software tools[1] that practitioners commonly use rely *entirely* on a manual approach, with which the user has to optimize the redistricting criteria with a trial-and-error approach. With such software, even expert users will need days or even weeks to manually construct a districting solution, which still may not be of sufficient quality in terms of

[1] There are many commercial redistricting software packages, such as: ArcGIS Redistricting Extension and Maptitude. There are also a number of web-based redistricting tools to allow users manually draw districts, such as Dave's redistricting site (http://gardow.com/davebradlee/redistricting), and Azavea's web-based redistricting (http://www.redistrictingthenation.com). These tools are all manual.

satisfying all criteria and constraints. For political redistricting, for example, each state or city in the U.S. usually has one or more full-time technicians, who often need months of preparation to generate just a few redistricting plans.

To address both the research challenge and practical need in solving real-world redistricting problems, this research develops a new spatial optimization method and a comprehensive system for redistricting. Specifically, my dissertation work achieves the following three objectives.

(1) Develop an efficient and effective spatial optimization method for redistricting. The developed method can optimize multiple criteria under multiple constraints and construct high-quality districting optimization solutions. The new optimization method is more efficient (being able to allow real-time user interaction), more flexible (considering multiple user-expressed criteria and constraints), and more powerful (in terms of optimization quality) than existing methods. The outcome of this task is evaluated by comparing with existing automated optimization methods with real-world case studies.

(2) Develop a redistricting framework and system, ***iRedistrict***, based on the new optimization method to integrate a variety of optimization criteria and constraints and provide user-friendly visual interface for incorporating domain knowledge, configuring optimization criteria and methodology parameters, and ultimately meet the needs of real-world applications. It is particularly useful for users of different skill levels, including researchers, practitioners, and the general public, and thus enables public participation in challenging redistricting tasks that are of immense public interest. Performance evaluations with real-world case studies are carried out. It potentially can enable broad

public participation in redistricting practices, which are currently inaccessible to the public.

(3) Extend the spatial optimization method to solve a different spatial optimization problem, i.e., spatial community structure detection in complex networks. Spatial community structure detection is to partition spatially embedded networks to reveal spatial communities by optimizing an objective function. Moreover, a series of new evaluations are carried out with synthetic datasets. This set of evaluations is different from the previous evaluations with case studies in that, the optimal solution is known with synthetic data and therefore it is possible to evaluate (1) whether the optimization method can discover the true pattern (global optima), and (2) how different data characteristics may affect the performance of the method.

This dissertation is organized into six chapters. Chapter 1 gives a brief introduction and summary of the dissertation work. Chapter 2 presents a comprehensive literature review. Chapter 3 introduces the new computational method for geographic redistricting problems. Chapter 4 evaluates the performance of the new method with real-world redistricting problems, presents the visual interface for user interaction, and introduces a set of computational solutions for handling large datasets in real applications. Chapter 5 presents a new application of the optimization method to detect spatial community structure detection in complex networks. Chapter 6 gives a conclusion for discussions on future work.

CHAPTER 2
RELATED RESEARCH

2.1 General methods for non-spatial combinatorial optimization

Algorithms for combinatorial optimization problems can be either exact or approximate. Exact algorithms are guaranteed to find an optimal solution, while approximate algorithms aim to find near-optimal solutions in a reasonable time. Since many combinatorial optimization problems are NP-complete, exact algorithms such as integer linear programming can only be used when the input data size for the problem is very small. Redistricting problems in real world are often too large for exact methods. Therefore in this section I focus on approximate algorithms that are based on metaheuristics. More complete discussion of algorithms for combinatorial optimization can be found in (Nemhauser and Wolsey 1999, Papadimitriou and Steiglitz 1998).

Metaheuristics are high-level strategies that use different heuristic methods to explore the search space and final near-optimal solutions (Blum and Roli 2003). Metaheuristics for combinatorial optimization include Genetic Algorithms, Ant Colony Optimization, Simulated Annealing, Iterated Local Search, and Tabu Search.

Genetic Algorithms (Holland 1975) encode a candidate solution of an optimization problem as a string. Through the evolution of a solution population (i.e., a set of strings), one can find better solutions, which are evaluated with a fitness function. For each generation, some individual solutions of the current population are selected to

generate a new population of solutions. The selected solutions are recombined through operations such as crossover and mutation of their string-based representation. The solutions with higher fitness values have more chance to be selected. This process is repeated until a certain stopping condition is met, and the best solution in the final population will be the final output.

Ant Colony Optimization (Dorigo, Caro and Gambardella 1999) is inspired by the behavior of ants and based on a parameterized probabilistic model. Stochastic solution construction procedures called artificial ants are often used, which iteratively add solution components to partial solutions based on information about a promising solution and previously acquired good solutions.

Simulated Annealing (Kirpatrick, Gelatt and Vecchi 1983) tries to solve optimization problems by simulating the physical annealing process. A simulated annealing method starts with an initial solution and a temperature parameter. At each step, the current solution is replaced with a random neighbor in the search space, with a probability that is a function of the temperature and the differences between their object function values. Such a process can escape local optima by allowing moves resulting in worse solutions with a probability, and the probability is decreasing along with the decrease of temperature.

Iterated Local Search (Lourenco, Martin and Stützle 2003) is based on the idea that iteratively builds a sequence of solutions to find better solutions. ILS starts with an initial solution and finds a local optimum with a local search such as hill climbing. Then it perturbs the solution and restarts the local search to find another local optimum. The

process is repeated until a certain stopping condition is met. The best among the local optimal solutions will be the final solution.

Tabu Search (Glover 1990) improves local search by using a short term memory (a Tabu list) to escape from local optima and avoid cycles. Tabu search finds a best move at each step even if the move is non-improving. A Tabu method keeps a list of objects that have recently been moved, which cannot move again. The list (called Tabu list) is a queue of a certain length (i.e., Tabu length k). Once an object is moved, it is inserted to the end of the queue. If the queue is full (i.e. having more than k objects), then the first object in the queue will be dropped and can move again. Periodically, the list is cleared and all objects can move (which is called restart). The Tabu search stops at a predefined condition such as the total number of moves or a maximum number of consecutive non-improving moves.

2.2 Specific methods for geographic districting

Methods for geographic districting can be generally divided into two main categories: divisive methods and agglomerative methods (Di Cortona et al. 1999). Divisive methods consider the space as a whole and divide it into different districts, while agglomerative methods consider the territory as a set of units and group the units into districts. Divisive methods include the successive dichotomies strategy (Forrest 1964) and the wedge-cutting strategy (Chance 1965). Agglomerative methods are more commonly used, and can be further divided into several major approaches: location-allocation methods, multi-kernel growth techniques, set-partitioning techniques, local search methods, and metaheuristics.

2.2.1 Methods based on location-allocation

In location-allocation methods, each unit is assigned to a territory center according to certain criteria and constraints, and then units assigned to the same territory center are grouped into a district (Hess et al. 1965). Kalcsics, Nickel, and Schröder (2005) combined a location-allocation method with optimal split resolution techniques, but the running time was too high to be practically useful. Segura-Ramiro et al. (2007) proposed a heuristic method based on location-allocation to solve a territory design problem for a beverage distribution firm. They extended the location-allocation method by Kalcsics, Nickel, and Schröder (2005) to handle contiguity constraint and multiple balancing constraints such as balancing the number of customers and sales volume. The method tries to minimize a dispersity measure to achieve compact districts. A local search was applied after an iteration of the location-allocation process to improve the dispersity measure. Experiments showed that this method could produce solutions of good quality but the execution time was long and the results were not comparable to those of other methods. Ko et al. (2015) integrated redistricting and location-allocation problems and used intra-district service transfer to address work overload problems.

2.2.2 Methods based on multi-kernel growth

Multi-kernel growth techniques first select some units as seeds, which gradually grow into districts. Vickrey (1961) selected one seed at a time and generated the next seed until a region was completed. A unit was first selected randomly as the reference area, and the unit farthest from this reference area was selected as the seed for a region.

Neighboring unassigned units were added to this region until a certain condition was met, and then the farthest unassigned unit from the reference area was selected as the next seed. This method is further studied in other researches (Gearhart and Liittschwager 1969, Openshaw 1977). Its main problem is that the condition for stopping the growth of a region is difficult to define and the quality of final regions are not sufficiently good for practical uses.

2.2.3 Methods based on set-partitioning

Set-partitioning methods first generate a large set of candidate districts that meet the required conditions such as contiguity, compactness and population, and then some candidate districts are selected based on an objective function to form a district plan (Garfinkel and Nemhauser 1970). Mehrotra et al. (1998a) developed a column generation algorithm to consider more candidate districts. Different criteria can be used to generate the candidate districts, but only a small number of units can be processed by set-partitioning methods due to the combinatorial complexity.

2.2.4 Methods based on local search

Local search methods try to improve an initial districting plan by moving units between neighboring districts to optimize the objective function of some criteria. Nagel (1965) proposed two types of moves: moving one unit at a time and trading units between two neighboring regions. Several conditions must be met in this process, such as spatial contiguity and the number of districts. Sammons (1978) allowed the non-improving moves in the process to escape from local optima. Yamada (2009) formulated

redistricting as a mini-max spanning forest problem, and used local-based search methods to solve it. Since these methods need an initial districting plan, they can be used to improve the plans generated by other approaches. Local search methods are relatively fast but less powerful in optimization since it can only search a small solution space.

2.2.5 Methods based on metaheuristics

Metaheuristics used for redistricting include simulated annealing, Tabu search, and genetic algorithms. Browdy (1990) proposed a simulated annealing method for redistricting to make non-improving moves with a certain probability. Huntley (1996) developed an algorithm based on simulated annealing for multi-objective service districting problems and used it to a school districting problem considering criteria of school utilization, efficient transport, and proximity of students to schools. Bergey, Ragsdale, and Hoskote (2003) used simulated annealing to improve the performance of a genetic algorithm. Rincon-Garcia et al. (2013) used a multi-objective simulated annealing algorithm to deal with the redistricting problem. Bennett (2010) studied the home healthcare nurse districting problem as a set partitioning model and combined column generation and local search to find solutions. The author argued that a major advantage of the method was easy adaptation for different scenarios such as different workload balance parameters and nurse team sizes. Joshi (2011) developed a constraint-based polygon spatial clustering algorithm for redistricting. The algorithm consists of three steps: select seeds, find the best cluster to grow, and find the best polygon to be added to this cluster. The algorithm was applied to the congressional redistricting problem and the school districting problem, and its performance was better than simulated annealing (Macmillan

2001) and genetic algorithms (Bacao, Lobo and Painho 2005) in terms of the criteria including equal population and compactness. Zhang and Brown (2013) used a method similar to the constraint-based polygonal spatial clustering algorithm to generate districting plans for police patrol.

Bozkaya (1999) developed an algorithm based on Tabu Search for political districting problems. The algorithm uses a region growing method to generate a starting solution, and then iteratively makes moves between neighboring districts based on the Tabu Search principles. A meta-heuristic called Probabilistic Diversification and Intensification could be integrated with the Tabu search to improve performance. Bozkaya et al. (2003) formulated the redistricting problem as a multi-criteria problem, and used a Tabu search and adaptive memory heuristic to solve the problem. Gonzalez-Ramirez et al. (2011) used a hybrid approach that combined the greedy randomized adaptive search procedure (GRASP) and the Tabu search to solve a districting problem of a parcel company. Assis et al. (2014) used a solution based on GRASP to address a multcriteria capacitated redistricting problem in power meter reading.

Xiao (2003, 2008) proposed a framework for the implementation of evolutionary algorithms for different geographical optimization problems such as redistricting. The framework used a graph-based representation to formulate different geographical optimization problems, and different algorithms are designed for initialization, recombination, and mutation operations in the evolutionary algorithm. This evolutionary algorithm was applied to the Iowa congress redistricting, and good solutions could be found. However, it is not as efficient and effective compared to other methods (Kim 2011). The author also suggested that problem-specific knowledge and heuristic methods

could be combined to improve the performance. Chou et al. (2012) used Interactive Evolutionary Computation (IEC) with Validated Surrogate Fitness functions to discover good redistricting plans for the Philadelphia City Council. Hu et al. (2014) developed a non-dominated sorting genetic algorithm to tackle a bi-objective model for the location and districting planning of earthquake shelters. Castelli et al. (2015) used geometric semantic genetic operators that employed semantic information directly in the evolutionary search process to improve its optimization ability for the electoral redistricting problem. Liu et al. (2016) developed a scalable evolutionary computational approach to use massively parallel high performance computing for political redistricting. Vanneschi et al. (2017) used a multi-objective genetic algorithm Pareto-based NSGA-II with a variable neighborhood search strategy to address the electoral redistricting problem.

The common challenge to these metaheuristics methods in solving spatial districting problems is to satisfy spatial constraints (such as contiguity) while exploring the solution space. One of the key contributions of the proposed methodology is to develop an efficient approach that can simultaneously guarantee spatial contiguity, computational efficiency, and effective searching strategies.

CHAPTER 3
A NEW COMPUTATIONAL METHOD FOR GEOGRAPHIC REDISTRICTING PROBLEMS

This Chapter introduces a new spatial optimization method for redistricting, which extends the traditional Tabu search heuristic with a novel strategy for enforcing geographic contiguity and an efficient way for defining, finding, and evaluating candidate moves. The new algorithm significantly improves both the efficiency and optimization quality for solving redistricting problems and thus enables automated or semi-automated solutions for real-world redistricting tasks. The algorithm is designed to address a wide range real world redistricting problems. I first review and categorize various criteria and constraints used in different real-world redistricting problems, and then develop a generic spatial optimization algorithm to flexibly incorporate different sets of criteria and constraints, with efficient and effective optimization strategies.

This Chapter focuses on the presentation of the core algorithms. Chapter 4 will present performance evaluations with real-world case studies, comparison with existing methods, visual interfaces for user interaction, and computational solutions for handling large datasets.

3.1 Criteria for redistricting

Existing research in the literature usually deals with different geographic districting problems separately and develops methods (or extensions) for each specific problem. This research categorizes and integrates different criteria to help develop a general framework for solving a wide range of redistricting problems. For political redistricting alone, Williams (1995) classified the optimization criteria into three types: demographic (e.g., equal population and minority representation), geographic (e.g., contiguity, compactness, and community integrity), and political (e.g., proportionality and similarity to the existing plan). However, this type of classification is not based on how each criterion is optimized. For example, equal population and minority representation are both demographic criteria but they require different strategies to optimize. This research classifies districting criteria based on how they can be optimized.

I surveyed the criteria, constraints, and objective functions used in different districting problems, extracted a set of commonalities and generic criteria, and developed an optimization framework based on the categorization to allow flexible combinations of criteria and thus meet the needs of different redistricting problems. Moreover, for each type of criteria, a specific optimization strategy can be developed to maximize the computational speed and optimization performance. Following are the categorized high-level groupings of redistricting criteria.

(1) **Geographic constraints**, including spatial contiguity, must-link constraint, cannot-link constraint, and fixed location constraint. Contiguity constraint requires that each district must be contiguous. Must-link constraint requires two

objects must stay in one district, while cannot-link constraint means the opposite. Many districting problems, especially service districting problems, require that each district contain exactly one fixed location. These constraints can be treated similarly, where the spatial connectivity are checked when a plan is initialized and when objects are moved between districts. Moreover, by exploiting such constraints, a more efficient strategy can be constructed to only explore the search space that satisfies the constraints.

(2) **Balance of district sizes**, such as equal population, equal household, balance of workload, and balance of the demand. These criteria require that a certain measure or variable value be nearly the same across all districts. To optimize such criteria, the optimization method can adopt specialized strategy to efficiently find candidate solutions such as trading units between districts and building indices to speed up such searches. This group of criteria can either be integrated in the objective function or treated as constraints (where the measure value must be within a certain range to a target value).

(3) **District-specific targets** such as majority-minority districts. This type of criteria is only evaluated for certain districts. For example, a majority-minority district is a district where a minority constitutes the majority of the voting age population in the district. There may be a required number of such districts for a specific redistricting task. Such criteria require that the optimization process be able to achieve different target values for different districts.

(4) **Global criteria** such as compactness, total workload, travel distance, similarity to the existing plan, and preserving the political boundaries. The uniqueness of such criteria lies in the fact that the solution is evaluated as a whole. This type of criteria potentially can be the performance bottleneck in the optimization speed because a local change has to be evaluated by looking at its global impact. For example, trading two units between two school districts may significantly change the short-path bus route in one or both. These criteria have no specific target for a district but a general target for the whole plan. They are usually integrated in the objective function.

(5) **Vague and subjective criteria** such as preserving communities of interest or neighborhood, where different users may have different understanding of "neighborhood" or "communities". Therefore, communities of interest or neighborhood usually cannot be clearly defined, and local knowledge is needed. To incorporate such criteria, a visual interface and user interaction are needed so that the user can choose or draw neighborhoods on the map and then the computational algorithms can consider those inputs. This is a process that integrates human judgments and computational algorithms.

For a specific task, a subset of criteria can be selected interactively and different optimization strategies are then integrated to achieve the best possible optimization quality and efficiency.

3.2 A new spatial optimization algorithm based on Tabu search

With the systematic categorization of different criteria summarized above, this research develops a new spatial optimization algorithm based on the Tabu search. The new algorithm guarantees geographic contiguity, achieves high efficiency, and at the same time significantly improves optimization quality over existing methods. The objective function consists of a set of user-selected criteria, each of which has an optional weight. The general steps of the new spatial optimization algorithm are shown in Algorithm 1, to give an overall understanding of the method. Details for each step and related algorithms will be explained in subsequent sections.

Algorithm 1: General Steps

1. **Initialization**—create an initial plan, by randomly portioning the space into a set of geographically contiguous regions;
2. **Optimization**—repeat the following steps until a stop condition is met:
   i. Find all candidate moves within the current solution;
   ii. Find the best move among all candidates, or the best switch of two candidate moves, according to an objective function;
   iii. Accept the best move or switch to modify the current solution, and update the best solution if the new solution is better;
3. **Output**—output the best solution recorded during the optimization.
4. **Repetition** (Optional)—repeat steps 1–3 to generate a set of alternative solutions, which the user can interactively examine and compare.

One of the major contributions of the new method is that it analyzes the contiguity relationship among objects in each district and efficiently identifies all possible moves along the border, including both single-object moves and multiple-object moves (as shown in Figure 3.1). Each move modifies the district boundary by moving an object (or multiple objects) to the neighboring district or switching objects between neighboring districts. Each move maintains all considered constraints such as geographic contiguity. Existing approaches can only allow single-object moves or switches while the new approach also allows multi-object moves.

	Single-Object Moves ***Multi-Object Moves***	*Pairs to Switch*
A→B	{1}, {5}, {14} {2, 1}, {10, 14}	10 pairs
B→A	{6}, {17} {11, 9, 17}, {9, 17}	
A→C	{14} {10, 14}	1 pair
C→A	{15} {21, 15, 23}	
B→C	{8}, {12}, {17} {11, 9, 17}, {9, 17}	18 pairs
C→B	{13}, {18} {16, 13}, {19, 13, 16, 20, 24}, {22, 15, 21, 23}, {21, 15, 23}	
Total:	**24 moves,**	**29 switches**

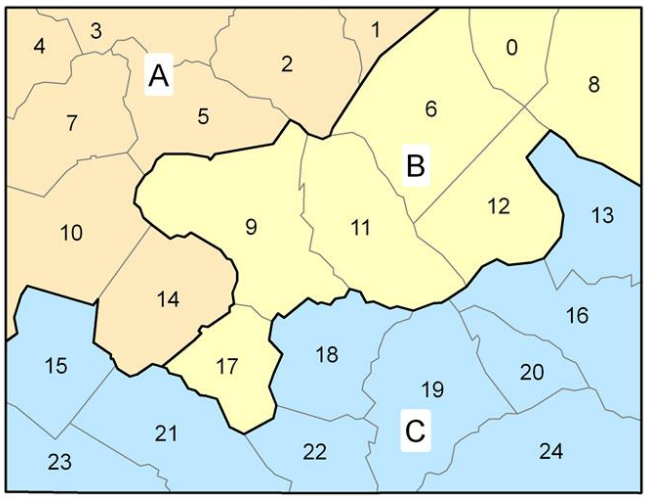

Figure 3.1 Multi-object moves under the contiguity constraint.

In the new approach, if an object on the border between two districts cannot move due to the contiguity constraint, the method finds a minimal set of objects that will move *together* with the object to maintain contiguity (Figure 3.1). In other words, in this new method, all objects on the border between two districts can move—some move by themselves and others move with two or more objects. For example, in Figure 3.1, if we

move the object (or polygon) 9 from district B to A, the contiguity of the district B will be broken. In the new method, object 9 and object 17 will move together. This new moving strategy is then combined with the Tabu search heuristics to enable a new optimization algorithm, as shown in Figure 3.2.

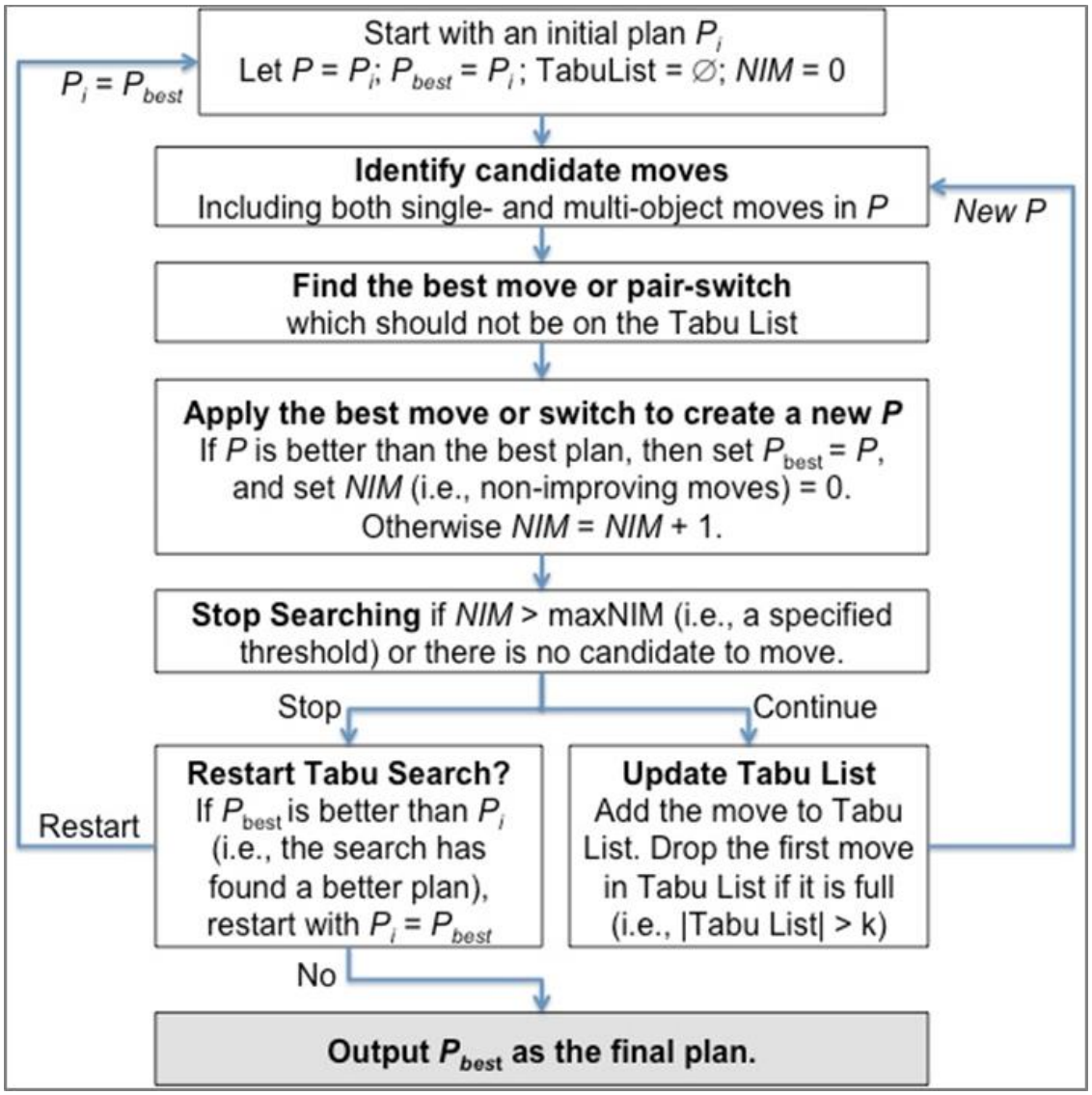

Figure 3.2: An overview of the Tabu-based optimization algorithm.

Figure 3.2 shows an overview of the new optimization algorithm, which is a Tabu search combined with the new contiguity-enforcing moving strategy. Tabu search methods have been used in many different applications and been shown to outperform alternative approaches (Glover 1990, Battiti and Bertossi 1999, Bozkaya et al. 2003). The algorithm progressively improves the quality of an initial plan by iteratively moving objects from one district to another. First, candidate moves are identified (including both single-object and multi-object moves). Second, the best move among them is identified and applied. The moved objects will be placed on the Tabu list for a certain period and cannot be moved again during that period, which is the key strategy in the traditional Tabu search heuristic. After each move, the list of candidate moves will be updated, and the best will be found and moved again. This process repeats until a stopping condition is met. Below I will explain the key steps in the algorithm in detail.

3.2.1 Initialization under contiguity constraint

To generate an initial redistricting plan, a simple seed-growing method is used, which randomly groups objects into r geographically contiguous districts (see Algorithm 2). First, r seeds (spatial objects) are selected randomly, each representing a district. Then districts grow one at a time by adding a non-assigned neighboring object to it. This process repeats until all objects are assigned to a district. Other initialization methods may also be used to generate an initial plan. The choice of an initialization method is not critical as long as it is a random process and can generate different plans when repeated. The initialization method should guarantee that each district is geographically contiguous.

Algorithm 2: Initialization under contiguity constraint

Input: *S*: a set of spatial objects, |*S*| = n;

C: a n*n contiguity matrix;

r: the number of districts, 1< *r* << *n*;

Steps:

1. Randomly select *r* objects from *S*, each being a district D_m, *m* = 1 .. *r*;
2. For each district D_m:
    a. Randomly select one of its unassigned neighbors *b* (if any);
    b. Assign *b* to D_m;
3. Repeat step 2 until all objects in *S* are assigned to a district.

3.2.2 Efficient algorithm for identifying multi-object moves

To efficiently identify all candidate moves (including both single-object moves and multi-object moves), I developed an efficient algorithm that can find all possible moves in linear time. Let us view the contiguity relations among spatial objects within a district as a graph G, where each spatial object is a node and two geographic neighbors are connected with an edge. If the removal of an object *u* from G cuts the graph into two or more disconnected components, object *u* is called an articulation point (a.k.a. cut point) in G. A bi-connected component is a maximal sub-graph of G that cannot be disconnected by deleting any object (Gabow 2000b). For example, the contiguity graph of district C in Figure 3.1 is shown in Figure 3.3, which has four cut points and five bi-connected components.

First, the algorithm finds all cut points and bi-connected components in each district with a depth-first search (DFS) method, which was first described in (Tarjan 1972), and later improved by (Gabow 2000b, Tarjan 1972). The complexity of the DFS algorithm is *O(n).*

Second, the algorithm identifies a multi-object move for each cut point. Figure 3.4 shows an example and Algorithm 3 shows the algorithmic steps. By definition, bi-connected components (BCCs) are connected only through cut points. If we view each BCC as a single "object", the contiguity graph becomes a spanning tree, with cut points as the connecting "edges". A BCC is a leaf in this tree if it only connects to one cut point (such as bcc_1 and bcc_5 in Figure 3.3). Since the removal of a cut point can cut a graph into two or more components, our strategy is to let the largest component represent the district and combine other components with the cut point to make a multi-object move. The size of a component can be defined as the number of spatial objects it contains or by other quantitative measures (such as the total population). The algorithm starts from a leaf BCC and traverses the tree from bottom up to find all multi-object moves. During the scan, the attribute values within a multi-object move are aggregated so that each multi-object move becomes a new "object". Aggregating information within a multi-object move speeds up the search for the best move since it does not need to visit all objects in each multi-object move.

The time complexity of Algorithm 3 is *O(n)*, where *n* is the number of objects in the district. Algorithm 3 is repeated for each district. Each non-cut point forms single object move and each cut point leads a multi-object move. Out of these moves, those on the border of two districts will be considered candidate moves. The same object (e.g.,

object 14 in A) may move to different neighboring districts (e.g., B or C), which are viewed as two different candidate moves. The list of candidate moves is updated after making a move (and thus creating a new plan), which is repeated many times in the optimization process (Step 2 in Algorithm 1).

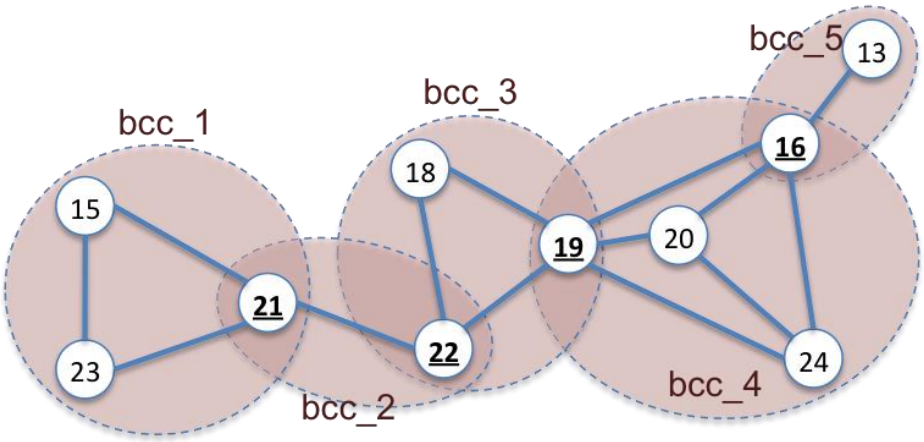

Figure 3.3 The contiguity relationship among the spatial objects in the district C in Figure 3.1. Neighbors are connected with edges, cut points are underlined, and dash-line ellipses show five bi-connected components.

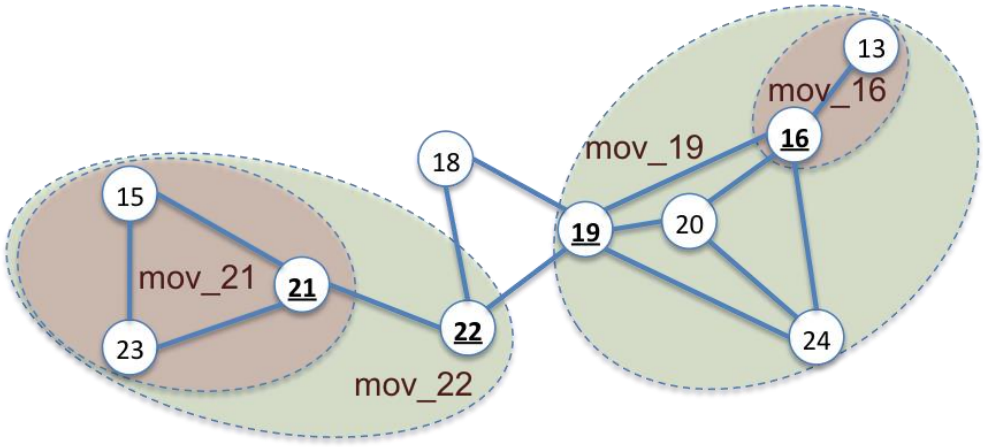

Figure 3.4 Composite moves (i.e., multi-object moves) for cut points. Each cut point will move as a composite move to maintain contiguity. For example, object 22 will move together with objects 15, 21, and 23 so that the remaining graph is still contiguous.

Algorithm 3: Identifying multi-object moves

Input:

S_d: the set of spatial objects in a district d;
C_d: a contiguity matrix of the objects in S_d;
A_d: attribute vector for each object in S_d;

Steps:

CompositeMoves = ∅; LeafBCC = ∅;

1. Find all cut points and biconnected components with DFS (S_d, C_d). (See (Gabow 2000a) for the details of the DFS algorithm.)
   bcc.CPT: the set of cut points that a biconnected component *bcc* contains;
   cpt.BCC: the set of biconnected components that a cut point *cpt* belongs to;
   cpt.maxC = ∅, which will keep the largest component for *cpt*;
   cpt.restC = ∅; which will keep the union of other components of *cpt*;
2. For each biconnected component *bcc*:
   If (|*bcc*.CPT| = 1): add *bcc* to LeafBCC;
3. Repeat the following steps until LeafBCC is empty;
   bcc = next biconnected component in LeafBCCs;
   cpt = the only cut point in *bcc.CPT*;
   i. If size(*bcc*)> size(*cpt*.maxC):
      cpt.restC = *cpt*.restC ∪ *cpt*.maxC;
      cpt.maxC = *bcc*;
      Else: *cpt*.restC = *cpt*.restC ∪ *bcc*;
   ii. Remove *bcc* from *cpt*.BCC;
   iii. If |*cpt*.BCC|=1 and size(*cpt*.maxC) < size(S_d)–size(*cpt*.maxC ∪ *cpt*.restC) + 1
      cpt.restC = *cpt*.maxC ∪ *cpt*.restC;
      bccR = the only remaining biconnected component in *cpt*.BCC;
      cpt.BCC = ∅;
      Remove *cpt* from *bccR*.CPT;
      If (|*bccR*.CPT| = 1):
         Add *bccR* to LeafBCC;
   iv. If *cpt*.BCC = ∅:
      cpt = aggregation of the vectors A_d in *cpt*.restC;
      Add *cpt* to CompositeMoves as a new composite move.

3.2.3 Efficient evaluation of candidate moves

Based on a given objective function *f*, each candidate move *m* is given a score δ_m, which is the difference in the overall objective value caused by the move. In other words, $\delta_m = f(P) - f(P_m)$, where P is the current plan and P_m is the new plan after making the move *m*. The move with the largest score is the best move (assuming the objective function is to be minimized). To achieve the best possible efficiency, the score for each move is calculated based on its *aggregated* attribute values and the aggregated information of the two involved districts. This strategy is called "dynamic scoring" (Altman and McDonald 2009). For example, given two districts A and B, and a set of candidate moves between them, the aggregated attribute values for each district may include its total population and dissolved shape boundary, which depend on the chosen set of optimization criteria. By aggregating data to districts it allows fast calculation of the score for each move without going through the entire dataset repeated and thus greatly improves efficiency. As such, it can calculate scores of all moves and find the best move in linear time.

3.2.4 Pair switch of candidate moves

Pair switches, i.e., exchanging two candidate moves between two neighboring districts, can be considered as combining two moves into one move, which are often needed to achieve better scores on certain criteria, such as equal population in redistricting (Bozkaya et al. 2003, Nagel 1965). Compared with existing methods that use pair switches, the pair switching in this research is unique and more effective since a switch can involve more than two objects. As shown in Figure 3.1, for example, the two

sets of polygons {2, 1} and {9, 17} can be switched to their opposite district. Not all pairs can be switched due to the contiguity constraint. For example, in Figure 3.1, object 14 and object 17 cannot be switched although each can move. This situation can be quickly identified by checking the following condition. Let M_1 and M_2 be two candidate moves, B_1 and B_2 be the boundary shared by each move with their destination district, respectively. Let B_s be the shared boundary between M_1 and M_2. If $B_1 \subseteq B_s$ or $B_2 \subseteq B_s$, then we cannot switch the two moves.

3.2.5 Efficient evaluation of pair switches

Evaluating pair switches can be time consuming if it enumerates and evaluates all possible pairs of moves. Based on the fact that pair switches are mainly used to optimize population equality, a new strategy is developed to efficiently find the best switch without enumerating all pairs. Suppose there are two districts A and B, each having a set of candidate moves. To find the best pair to switch, the moves in each district are sorted by their population. Then, given a move *u* in A, its population, and the population of A and B, we can calculate the target population of an ideal move in B to switch with *u*. Since the moves in B are already ordered, with a binary search we can quickly locate the move *m* in B with a population that is closest to the target population. We then search a certain number of moves on both sides of m in the order to find the "best" move *v* to switch with *u* in terms of the overall objective function. Note that this "best" move is for paring with *u* only. Repeat this for each move in A, we can get the best switch between A and B. The time complexity for evaluating pair switches is $O(n\log n)$, where *n* is the

number of moves in A and B. The best move identified in Section 3.2.3 is compared with the best switch identified here to determine which the overall best move is.

3.2.6 A new Tabu search algorithm

What makes the Tabu search heuristic unique is its short-memory strategy to avoid repeating the search paths that are already investigated and thus may force the search to escape local optima. Specifically, the search process uses a Tabu list to remember the most recent moves, which are prohibited to move again until they are removed from the list. The length of the Tabu list (k—the number of prohibited moves) is normally much smaller than the data set size (n). In our experiments, $k = 0.08n$. A Tabu search allows non-improving moves, i.e., it is acceptable that the best move does not improve the objective value. By allowing non-improving moves, it hopes to escape a local optimum and eventually found a better solution. The search stops when the number of consecutive non-improving moves exceeds a threshold (maxNIM). In our experiments, we set maxNIM = $3n$.

By changing several parameters, we can easily convert the algorithm in Figure 3.2 to two other trajectory-based optimization methods: the local greedy search (hill climbing) and the Kernighan–Lin algorithm. If $k = 0$ (i.e., no tabu) and maxNIM = 0 (i.e. does not allow non-improving moves), it becomes a local greedy search method. Local greedy search only accepts improving moves and stops at a local optimal. It is fast but often poor in optimization quality. If we set $k = \infty$ and maxNIM = ∞ (i.e. each move can move once and only once—in this case the search stops when there is no valid move), then it is the same as the Kernighan–Lin algorithm, which was originally developed for

graph partitioning (Kernighan and Lin 1970) and has been used in many optimization problems applications such as complex network analysis (Newman 2006a).

Moreover, by turning on and off new our contiguity-enforcing approach (which allows multi-object moves, as explained in Section 3.2.2), the algorithm presented in Figure 3.2 can be configured to become six different methods, as summarized in Table 3.1. If multi-object moves are not allowed (i.e., without our new approach), we have three traditional trajectory-based optimization methods: local greedy search, Kernighan–Lin (K-L) algorithm, and Tabu search. If our new contiguity-enforcing approach is integrated to allow multi-object moves, we have three new optimization methods: Greedy*, K-L*, and Tabu*, where the star (*) indicates the capability of multi-object moves. Our experiments show that each of the three new methods significantly outperforms its traditional version by a large margin and yet remains efficient.

Table 3.1 Different optimization methods.

	Multi-object moves	Tabu List Length (k)	Maximum number of consecutive non-improving moves (maxNIM)	Time Complexity
Greedy	No	k = 0	maxNIM = 0	$O(mn\log n)$, m<<n
K-L	No	k = ∞	maxNIM = ∞	$O(mn\log n)$
Tabu	No	k << n	maxNIM = 3n	$O(mn\log n)$
Greedy*	**Yes**	k = 0	maxNIM = 0	$O(mn\log n)$, m<<n
K-L*	**Yes**	k = ∞	maxNIM = ∞	$O(mn\log n)$
Tabu*	**Yes**	k << n	maxNIM = 3n	$O(mn\log n)$

3.2.7 Computational complexity

The overall complexity of the optimization method is $O(mkn\log n)$, where m is the number of criteria considered, k is the number of iterations during Tabu search, and n is the number of spatial units. Since m is generally small, k and n are the determining factors.

3.3 Optimization strategies for different types of criteria

Different types of criteria can be used in the spatial optimization algorithm introduced in Section 3.2. A specific redistricting task may consider multiple criteria and give each criterion a weight. The objective function f is the weighted combination of measures of the selected criteria:

$$f = \sum_{c=1}^{k} w_c m_c \tag{3.1}$$

where w_c is the weight for criterion c, m_c is the measure of criterion c, and k is the number of selected criteria. Note that the measures are all transformed and normalized so that a smaller measure value means a better quality. One of the main steps in the optimization process is to find the best move among the candidate moves based on the objective function. To achieve an overall efficiency for the algorithm, different types of criteria may need different optimization strategies, which I will explain below.

3.3.1 Geographic constraints

Geographic constraints are not in the objective function but are maintained and checked during the optimization process. To enforce spatial contiguity, a contiguity matrix is created. In the initialization process, the contiguity matrix is used to construct

an undirected graph. Each unit is a vertex, and two neighboring units are connected by an edge. The contiguity graph is used in the whole optimization process to make sure the generated regions are contiguous. Particularly, the identification of candidate moves, as introduced in Section 3.2, heavily rely on the contiguity graph to achieve high efficiency in finding all possible moves in a linear time.

3.3.2 Balance of district sizes

Balance of district sizes is one of the most common criteria for redistricting problems. In optimizing measures for balanced sizes, the pair switching strategy in Section 3.2.3 is very important. The balance of district sizes is normally measured by a "deviation" (Dev) value—the sum of absolute differences between each district's actual size (p_i) and its ideal size, which is the total size (P) divided by the total number of districts r.

$$Dev = \sum_{i=1}^{r} \left| p_i - \frac{P}{r} \right| \tag{3.2}$$

In some redistricting problems, the ideal size can be different for each district. For example, in school redistricting, the ideal size depends not only on the total student population and the total number of districts, but also on the capacity of each school (c_i).

$$Dev = \sum_{i=1}^{r} \left| p_i - \frac{c_i P}{\sum_{i=1}^{r} c_i} \right| \tag{3.3}$$

Multiple sizes can be considered at the same time, and each size can be assigned a weight. For example, in school redistricting, the ratio of the number of students to the capacity should be considered for different grades, and the deviation measure can be calculated as follows:

$$Dev = \sum_{i=1}^{r} \sum_{g=1}^{m} w_g \left| p_{ig} - \frac{c_{ig} P_g}{\sum_{i=1}^{r} c_{ig}} \right| \quad (3.3)$$

where g is the grade, m is the total number of grades, and w_g is the weight for grade g.

3.3.3 District-specific targets

Some criteria are only evaluated for certain districts. For example, in political redistricting, it is required for certain states (e.g., South Carolina) that there must be one or more majority-minority districts, in which the minority groups (e.g., Black population) make up a majority of the population. So the percentages of different racial groups will only be evaluated for some of the districts, and different targets can be set for different districts. In the initialization process, the targets are set for districts whose initial measures are close to the targets. The optimization algorithm calculates the measures for only the districts where the targets are set.

3.3.4 Compactness

Certain redistricting tasks require that the shape of each district should be as compact (or simple) as possible. For example, as a means to prevent gerrymandering, the constitution of Iowa specifically requires that the political redistricting process must consider compactness of each district. One commonly used compactness measure is the Polsby-Popper index, which divides the area (α_i) of the district by the area of a circle with the same perimeter (ρ_i) as that of the district (Polsby and Popper 1991). This measure ranges from 1 (perfect circle) to zero. This measure can be part of the overall objective function. There is no specific optimization strategy for the compactness measure.

$$Compactness = \frac{1}{r}\sum_{i=1}^{r}\frac{4\pi\alpha_i}{{\rho_i}^2} \quad (3.4)$$

3.3.5 Travel distance

For redistricting problems such as school redistricting and business service area redistricting, travel distance is an important factor to be considered. For example, the average travel distance to school needs to be minimized for school redistricting. The travel distance is calculated for every pair of the unit and the fixed location (e.g. school), and a distance matrix is created. The average travel distance is constantly updated using the distance matrix in the optimization process. Since the average distance can be affected by a few large distances, an average distance *order* measure can be used as a proxy. The distances from all units to a fixed location are ordered, and each distance is assigned an order number starting from 1. The optimization algorithm tries to minimize the average distance order measure, and tries to assign a unit to its closest fixed location.

3.4 Conclusion

This chapter presents a new computational method for geographic redistricting problems. Different criteria for different redistricting problems are reviewed and categorized. Based on the categorization, a new spatial optimization algorithm is developed to flexibly incorporate different sets of criteria and constraints. This new spatial optimization algorithm integrates the Tabu search with a new contiguity-enforcing approach that allows multi-object moves. It can guarantee geographic contiguity, achieve high efficiency, and at the same time significantly improve optimization quality over existing methods with innovative search strategies.

CHAPTER 4
PERFORMANCE EVALUATION, USER INTERACTION, AND COMPUTATIONAL SOLUTION FOR LARGE DATASETS

The key contributions of the new spatial optimization method include (1) computational efficiency—the new optimization strategies significantly improve the computational efficiency of existing methods and thus enable applications with redistricting optimization while allowing real-time user interaction, (2) optimization quality—the method reliably achieves much higher optimization quality than existing methods, and (3) flexibility—with both efficiency and quality the method provides a flexible framework to consider different sets of criteria and produce results that meet practical needs.

This Chapter presents a series of performance evaluations of the new method with real-world case studies and comparisons with existing methods. This Chapter also presents a redistricting system, ***iRedistrict***, which is based on the new method and has visual interfaces that allow users to define subjective criteria (e.g., community of interest), interactively configure optimization criteria and parameters, and visually evaluate, explore, and manage optimization outcomes. Lastly, this Chapter presents a set of computational solutions for handling large datasets in real applications. Combinatorial optimization problems are computationally demanding and most existing optimization methods can only deal with very small datasets to produce acceptable results. While the

new method presented in this dissertation is already significantly faster than existing methods, it still needs further computational solutions to handle large datasets, such as tens of thousands spatial objects, to produce optimization results that meet practical requirements.

4.1 Performance evaluation with case studies

Redistricting problems are encountered in many different application domains including political districting, school districting, sales districting, and community structure detection. The primary difference among these application problems is the set of criteria and constraints being considered in the optimization process. The new method introduced in Chapter 3 can consider different sets of criteria and constraints. I carried out several case studies to demonstrate and evaluate the method.

4.1.1 Iowa congressional redistricting

The criteria for congressional redistricting, established by the Iowa Constitution, include population equality, geographic contiguity, compactness, and respect for political subdivisions (county boundaries). It does not require majority-minority districts, since the minority group in Iowa is not sufficiently large. The Iowa state has 99 counties, which are to be divided into four congressional districts after the 2010 census. The total population of Iowa is 3,046,355. The Iowa Code also states that areas are not considered contiguous if they only meet at the points of the adjoining corners. In other words, two areas are considered contiguous if and only if they share at least a common border segment. Figure 4.1 shows the 99 counties with their population values labeled.

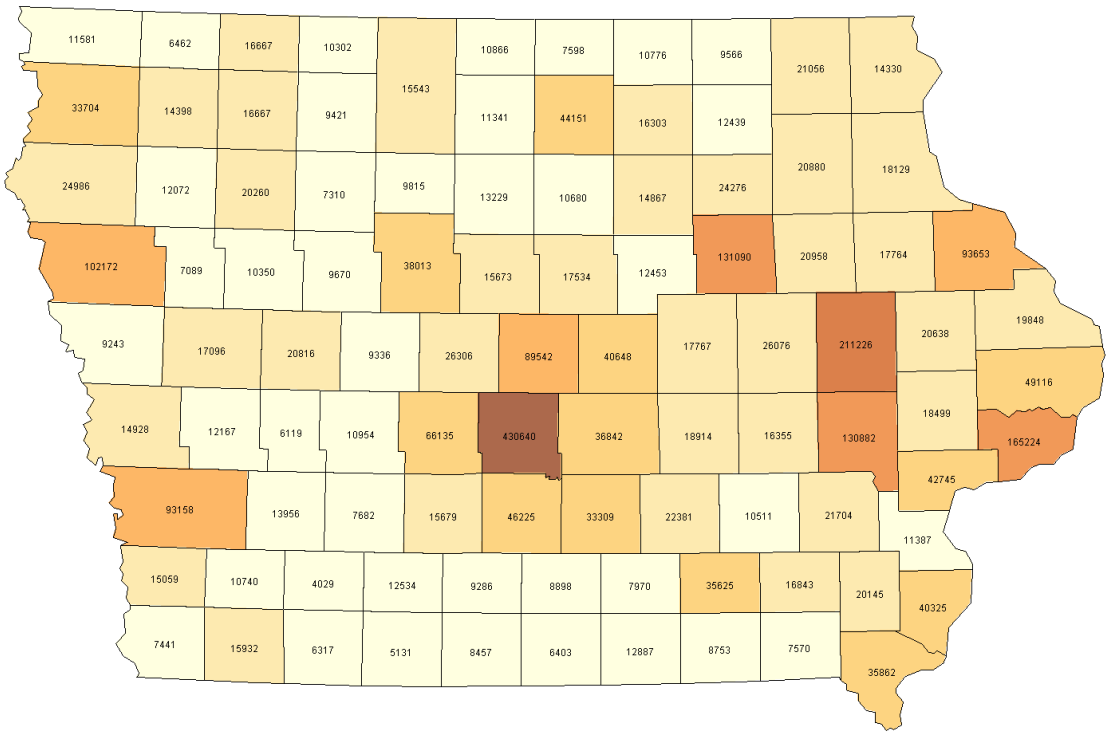

Figure 4.1 Iowa counties and their population (2010 census).

1) Optimizing population equality only

The first experiment considers only the population equality criterion. Population equality is measured with a "deviation" value (PopDev), which is the sum of absolute differences between each district's actual population and its ideal population, which is the total size divided by the total number of districts. For Iowa, the ideal population for each district is 761588 or 761589 (as 3,046,355 / 4 = 761588.75, which is used internally in the algorithm as the ideal population).

The six methods, as introduced in Chapter 3, are included in this experiment. The Greedy (local greedy search), K-L (Kernighan–Lin), and Tabu methods are three

optimization methods, while Greedy*, K-L*, and Tabu* are the new optimization methods developed in this research with the new contiguity-enforcing and optimization strategies. Specifically, Tabu* is the chosen new method in this research as it consistently achieves the best performance across all experiments.

Each method generates 1000 plans on an i7-3770 (3.40 GHz) machine. The summary statistics of the 1000 PopDev values for each method are shown in Table 4.1, which show that each of the new optimization methods significantly outperforms its traditional version. Particularly, the Tabu* method (i.e., the new optimization method of this research) reliably achieves the best performance, with average = 133 and standard deviation = 68, which statistically outperforms all other methods. The best plan of the 1000 results found by Tabu* by only optimizing the population equality criterion has a PopDev value of 4.5, which is probably the global optimal solution. The theoretical global optimal value for PopDev is 1.5 since the ideal population (761588.75) is not a whole number. Figure 4.2 shows the map for this plan. The computational time for Tabu* in this experiment is 33 seconds for generating 1000 plans, i.e., it takes 0.03 second to optimize each plan.

The standard deviation for the Tabu* method is 68, which is much smaller than all other methods. This indicates that performance of the Tabu* method is robust. With different random initializations, the method can always reliably reach a high-quality solution. This is important for several reasons. First, it shows that the optimization method can effectively escape local optima. Second, in practice, we do not need to repeatedly run the method many times in search of a good solution and thus it becomes possible for real-time and interactive use.

Table 4.1 Evaluations with Iowa data for optimizing population equality (PopDev).

1000 Runs	Traditional optimization methods			New methods		
	Greedy	K-L	Tabu	Greedy*	K-L*	Tabu*
Min	646	421	109	81	33	4.5
5%	3937	1432	650	559	165	51
Q1 (25%)	7037	3184	1364	1400	369	89
Median (50%)	**9531**	**4684**	**2149**	**2697**	**579**	**133**
Q2 (75%)	12546	6436	3590	4881	976	181
95%	18372	9092	7020	11036	1854	253
Max	169308	87299	70167	70167	8224	497
Standard Deviation	**8051**	**4077**	**3066**	**4611**	**668**	**68**

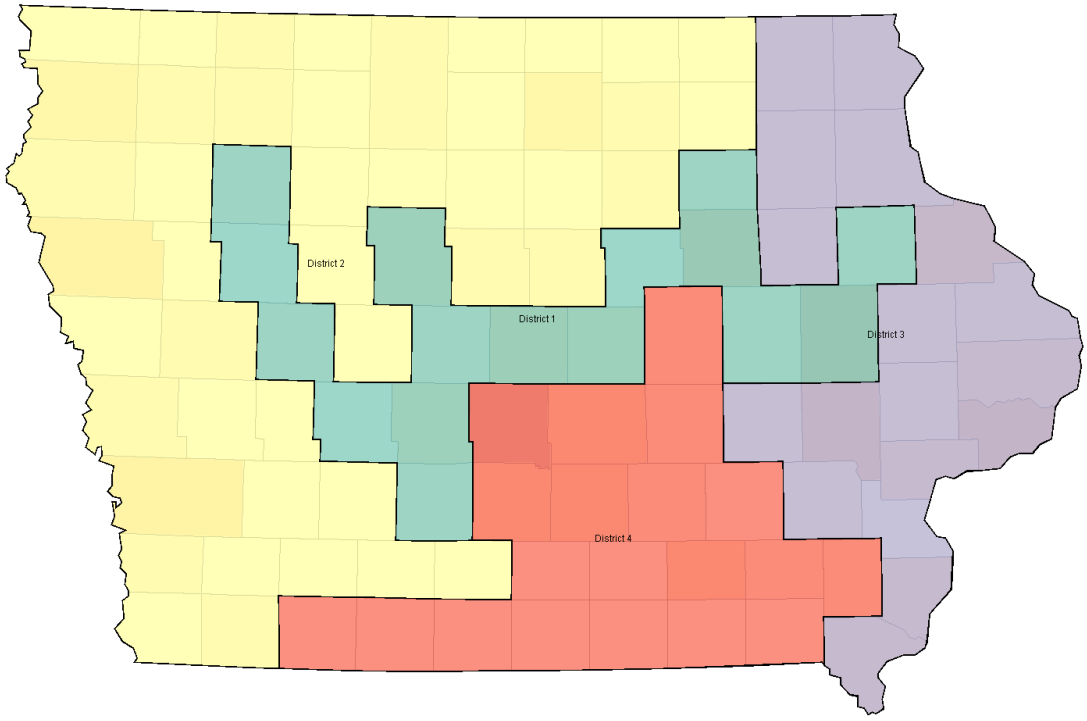

Figure 4.2 An Iowa plan of four districts with a population deviation (PopDev) of 4.5.

2) Optimizing both population equality and shape compactness.

The second experiment considers both population equality and compactness. As explained in Chapter 3, the compactness is measure with the Polsby-Popper index, which divides the area of the district by the area of a circle with the same perimeter as that of the district (Polsby and Popper 1991). This measure ranges from 1 (perfect shape) to zero.

In this experiment we focus on the difference between Tabu and Tabu*. Each method generates 1000 plans on an i7-3770 (3.40 GHz) machine. The summary statistics of the two sets of measure values for each method are shown in Table 4.2. Note that a larger value for compactness means a more compact shape, while a smaller value for PopDev means better population equality. Internally in the algorithm, these measures are transformed and normalized before being combined into an objective function. The results show that Tabu* again significantly outperforms its traditional version.

Table 4.2 Evaluation with Iowa data, optimizing population equality and compactness

1000 Runs	Tabu		Tabu*	
	PopDev	Compactness	PopDev	Compactness
Min	141	0.137	15	0.181
5%	802	0.169	79	0.219
Q1 (25%)	1678	0.201	155	0.256
Median (50%)	**2634**	**0.230**	**226**	**0.284**
Q2 (75%)	3987	0.265	319	0.318
95%	8198	0.317	479	0.378
Max	113794	0.407	1202	0.458
StdDev	4788	0.047	128	0.046

4.1.2 South Carolina congressional redistricting

The criteria for congressional redistricting in South Carolina include population equality, contiguousness, compactness, majority-minority districts, and communities of interest. County boundaries, municipality boundaries, and voting precinct boundaries should be considered when practical and appropriate, since they are considered as one kind of evidence of communities of interest. In this research, voting precincts are used as the spatial units. South Carolina has 2122 voting precincts (Figure 4.3), which are to be divided into 7 congressional districts based on 2010 census data. The total population of South Carolina is 4,625,364 and therefore the ideal population for each district is 660,766. In calculating the PopDev measure, the value 4,625,364 / 7 = 660766.285714 is used. The theoretical global optimal value is 2.857142.

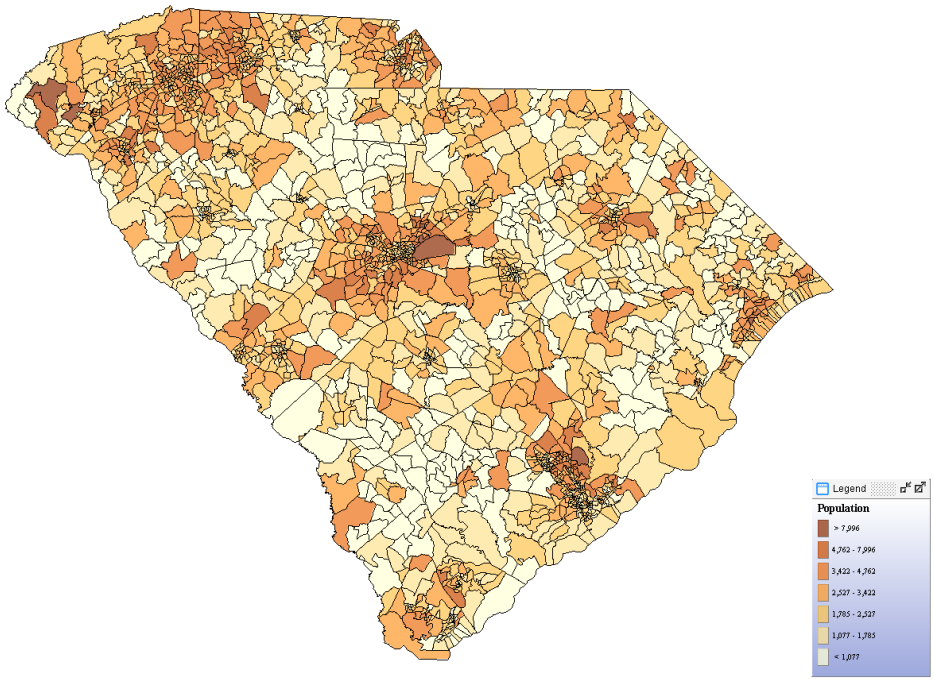

Figure 4.3 Population of South Carolina voting precincts (2010 census).

1) Optimizing population equality only

The South Carolina data set is much larger than the Iowa data set in terms of the number of spatial objects (units) and thus requires more computational time. However, more spatial objects actually make it easier to find the global optimum when only considering population equality. The best results from 1000 runs of the three traditional optimization methods are comparable with those of the new optimization methods integrated with the new methods, and except Greedy, all methods found many solutions to achieve the theoretical global optimal value. The new methods are more significantly more robust and consistent, evidenced by their very small standard deviation values. It is interesting to notice that the K-L method (which can be considered a special case of Tabu with a Tabu list of infinite length) and its new extension K-L* slightly outperform Tabu and Tabu*, respectively. This provides important insights on the configuration of Tabu parameters in relation to data size, which is a future direction for this research.

Table 4.3 Evaluations with South Carolina data, optimizing population equality only. Values are rounded a whole number or keeping one decimal digit for values less than 10. The theoretical global optimal value is 2.857142, which is rounded to 2.9 in the table.

1000 Runs	Traditional optimization methods			Combined with our approach		
	Greedy	K-L	Tabu	Greedy*	K-L*	Tabu*
Min	4.3	2.9	2.9	4.3	2.9	2.9
5%	20	2.9	2.9	14	2.9	2.9
Q1(25%)	61	2.9	4.8	46	2.9	4.3
Median (50%)	**305**	**4.3**	**8.3**	**113**	**4.3**	**6.3**
Q2(75%)	222392	5.7	16	364	5.7	9.8
95%	753547	13	20369	352285	9.4	17
Max	1744267	966028	1510368	1167422	54	120
StdDev	268679	77558	97871	137487	3	13

2) Optimizing population equality, compactness, and majority-minority districts

In this case study, three criteria are considered: population equality, district compactness, and creating a majority-minority district in which the minority population is the majority. In order to create a majority-minority district, a community of interest (COI) is outlined on the map by the user, which contains precincts with a large percentage of minority population (Figure 4.4). The optimization method will take this COI as input, optimize a district containing the COI to generate a majority-minority district, and in the meantime optimize all chosen criteria to generate a plan of seven districts.

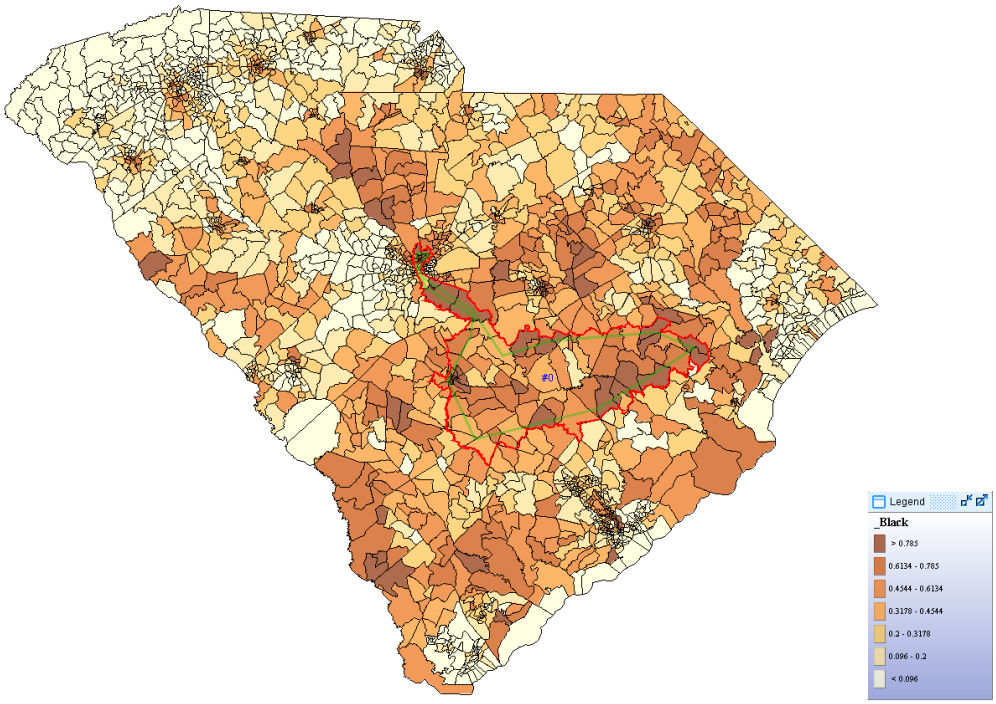

Figure 4.4 A user-drawn community of interest (COI).

Figure 4.5 shows an outcome plan, in which the PopDev value (69) is very good, each district has a compact shape, and there is a majority-minority district (with 53.52% minority population). In addition to the optimization quality and efficiency, another

advantage of the new method is that it is flexible to consider various criteria, allows interactive user inputs and visual inspection (which will be elaborated in Section 4.2).

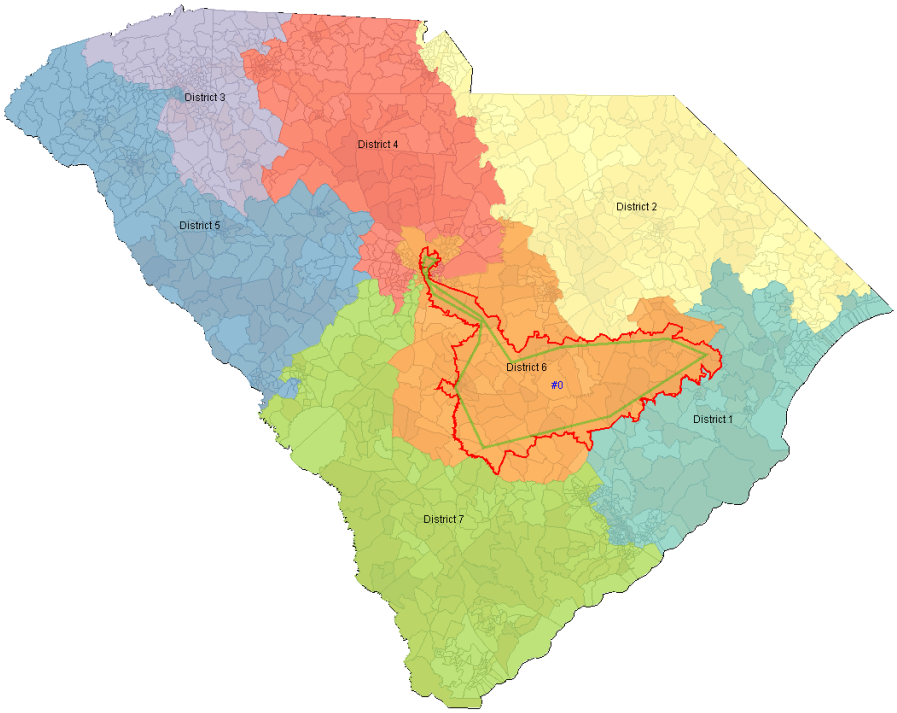

Figure 4.5 A plan with a majority-minority district.

4.1.3 School redistricting

School redistricting is different from political redistricting in two important aspects. First, the optimization criteria to be considered are quite different. Common criteria and constraints considered in school redistricting are listed below, among which the first two are considered as constraints. Second, each district is constructed around a fixed location, i.e., school. This location is not only important for certain criteria such as distance to school but also critical in determining the district boundary, which should contain the location.

- Spatial contiguity

- Each district contains one and only one school
- Balance the number of students to school capacity (for each grade)
- Shape compactness
- Average distance to school
- Existing school district boundaries

In this case study, we use a real-world scenario. Prince William County, Virginia has used the redistricting method and system developed in this research to redraw the boundaries of the school districts for its16 middle schools (Figure 4.6). All the six criteria listed above are used. Particularly, the projected student populations for each grade in future years are considered in evaluating the balance between student population and school capacity for each district. The choice, configuration, and weight of each criterion can be set interactive with visual interfaces (which is introduced in Section 4.2).

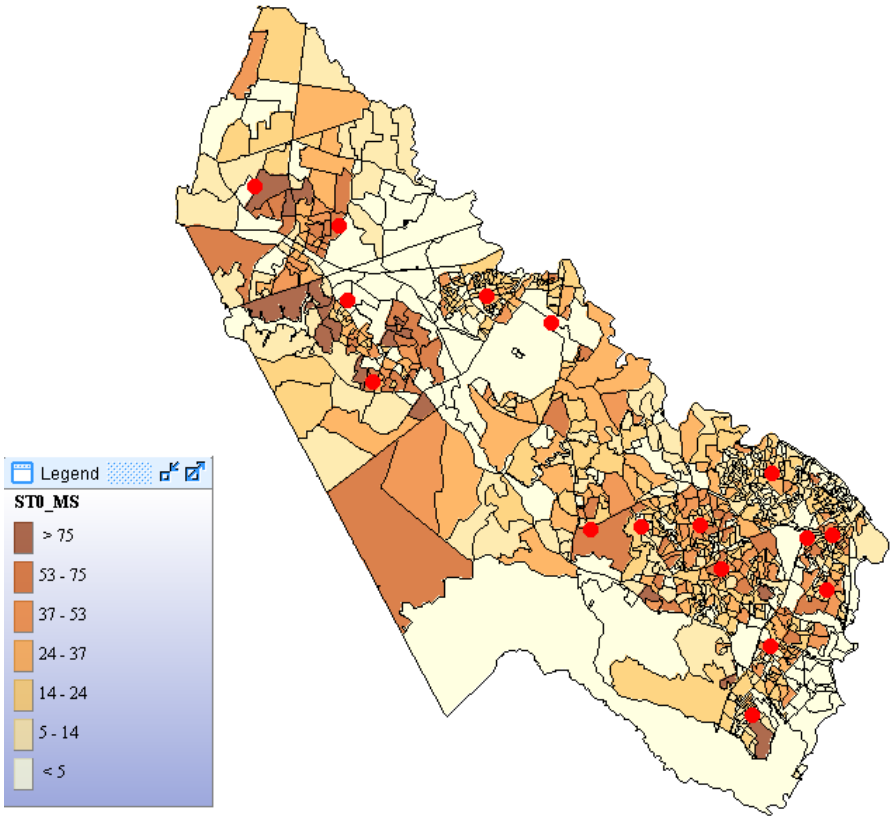

Figure 4.6 Middle school student enrollments in Prince William County, Virginia

Figure 4.7 shows one of the school redistricting plans, which achieves very good scores across the chosen criteria, much better than one could achieve with a manual approach as used in most of the current practices of school redistricting. Most importantly, the new method and its implemented system give general users the immense power to construct redistricting plans and participate in the redistricting process, which is impossible with existing methods and available software tools.

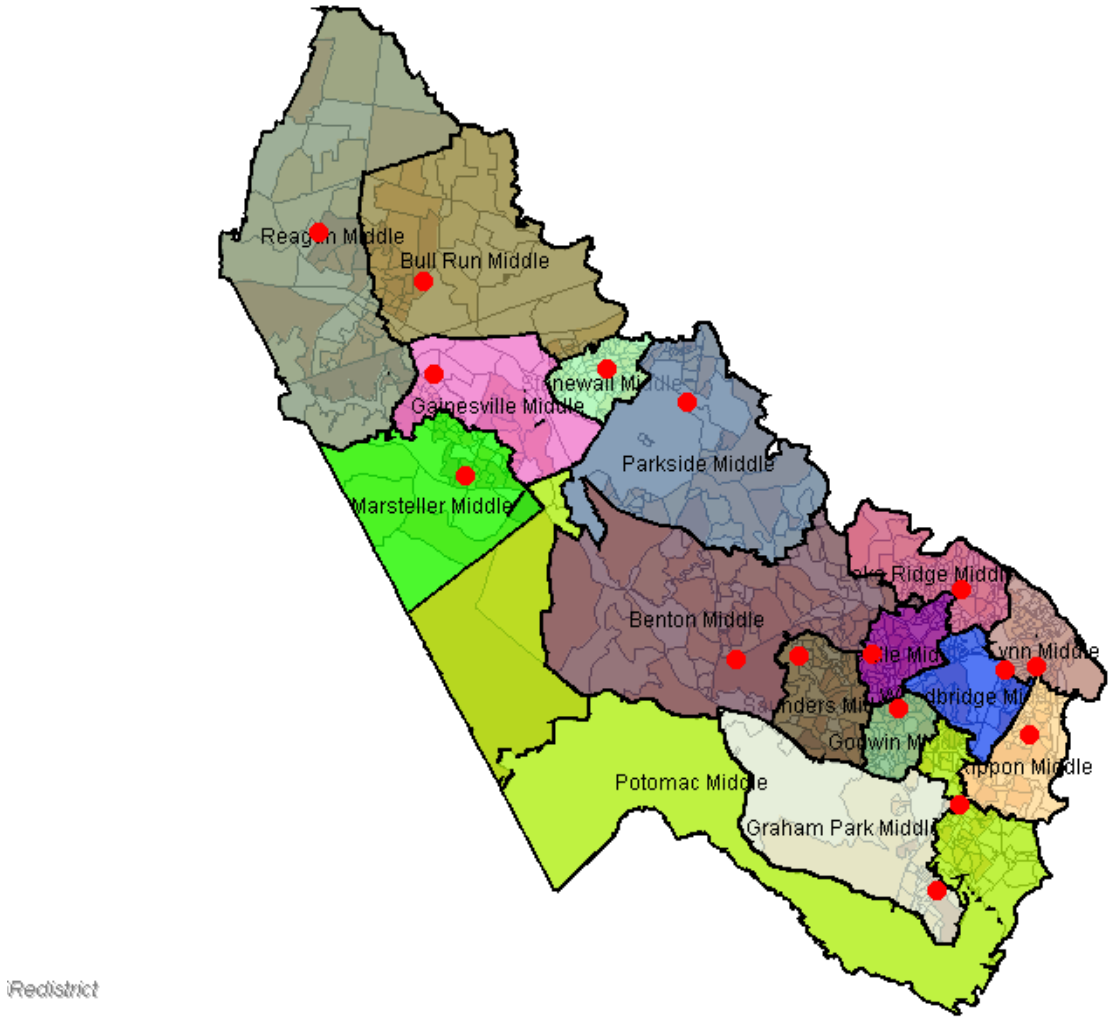

Figure 4.7 A school redistricting plan for middle schools in Prince William County.

4.2 Visual interface and user interaction to integrate human inputs

4.2.1 Subjective criteria

There are vague and subjective criteria that cannot be clearly defined, such as preserving communities of interest. Different people may have different understandings of "communities", for which local knowledge is needed. For such vague and subjective criteria, visual interface is needed to dynamically integrate human judgments with the computational method. For example, the user may draw several areas to indicate communities of interest to be preserved. Then the algorithm will optimize selected criteria under these constraints, i.e., each user-drawn community will not be split during the Tabu search. Figure 4.8 (Maps D to F) shows three selected results with such user provided constraints. For example, Map D and Map E are two different plans for the same set of user drawings, while Map F is a plan for a different set of drawings.

The results in Figure 4.8 are for the 2000 census data, which clearly show the capability and potential of the new method, with the ability to integrate user inputs on the fly. The population deviation value of each plan in Figure 4.8 is far below 0.1% of the total population. The compactness values of four plans (C-F) that considered shape are all better than (or at least equivalent to) that of the official plan in Iowa (2000-2010). Moreover, this is done without much technical challenge or investment of time for the user. With existing redistricting software, even a technical person or expert may need several days to construct just one plan of a similar quality. The algorithm can easily consider more criteria in the optimization process, which are introduced in Section 3.1.

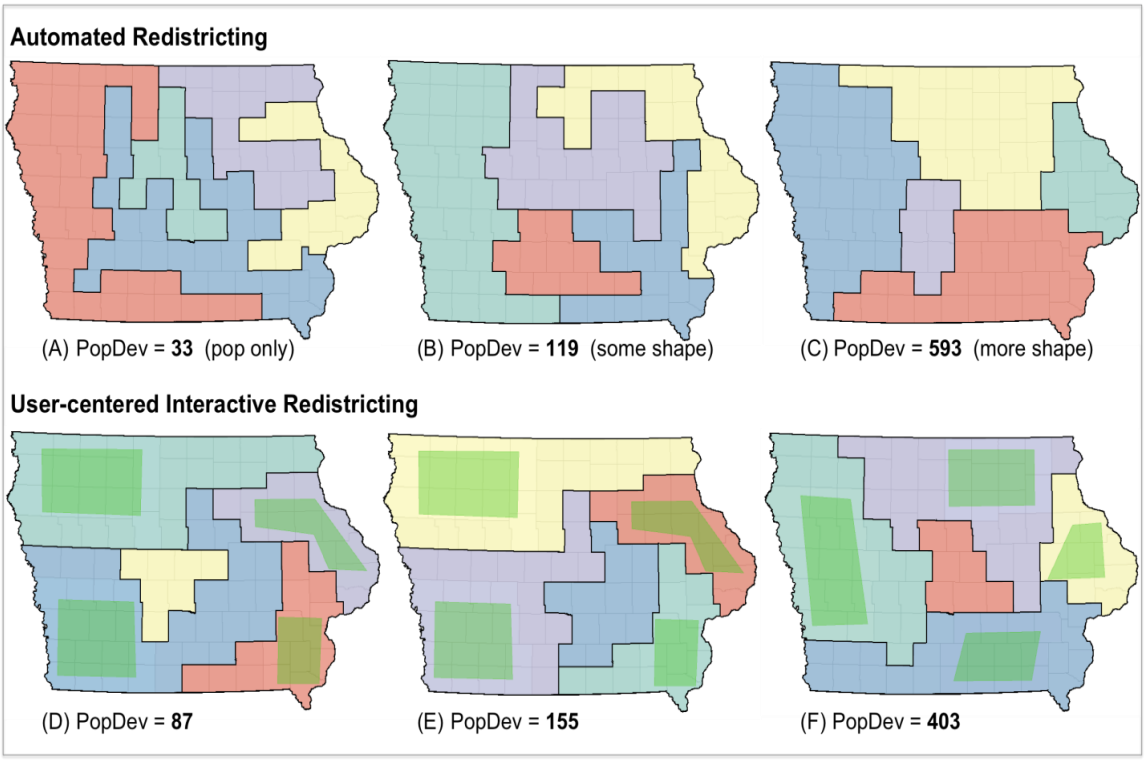

Figure 4.8 Iowa congressional redistricting with the 2000 census data. Maps (A-C) show three selected results with the new method without user drawing. Maps (D-F) show three selected results with user drawings (indicated by the green semi-transparent areas). PopDev is the measure for equal population, which is the total deviation between district population and its target population, which is the smaller the better.

4.2.2 Visual interface and user interaction

The overall visual interface for the optimization method is shown in Figure 4.9. The user can choose criteria to be used, configure the parameters for each criteria, set parameters for the optimization method (such as the number of districts in each plan and the number of plans to be created), visually examine the criteria scores of outcome plans in a scatterplot, view a specific plan in the map, and specify preferences for vaguely defined criteria (such as communities of interest) with direct drawing on the map.

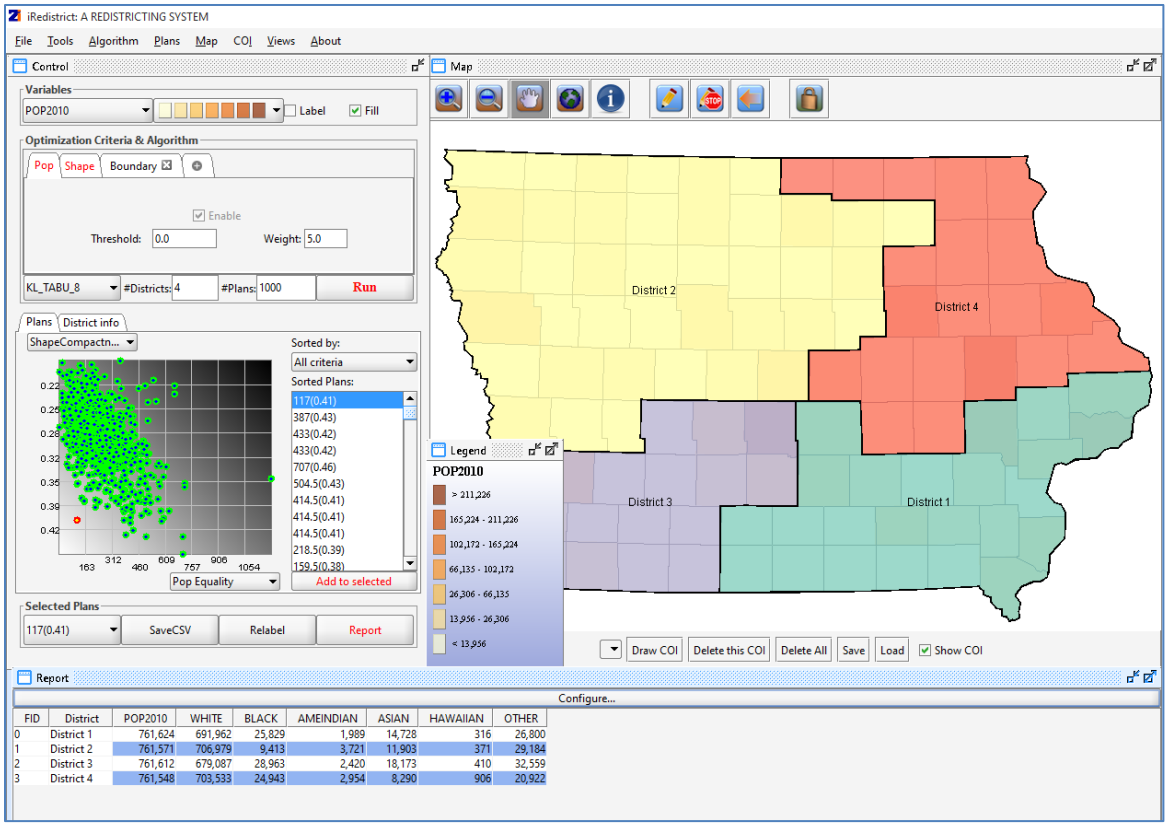

Figure 4.9 The redistricting system, *iRedistrict*, based on the new optimization method.

Figure 4.10 summarizes the analytics process for redistricting, including five interacting components: (A) data mapping, (B) user drawing to express constraints, (C) configuration of the optimization algorithm, (D) visual examination and comparison of alternative plans, and (E) managing desirable plans that are accumulated through an iterative process (Guo and Jin 2011a). The algorithm can run many times to create a set of alternative plans, with each run starting with a random initial plan. The user can visually examine the alternative plans to find the best and add them to a list of selected plans. If the alternative plans are not good enough, the user can use tools such as "edit" and "lock" to improve the plans. Through such an interactive and iterative process, the user can quickly obtain a set of high quality plans.

1) Mapping

The user can choose the variable to be classified for the unit layer and create a choropleth map. The number of classes and the color scheme can also be defined. The labels can be shown if needed. A legend panel is shown to display the color and the break for each class. This map can help the user understand the distribution of the selected variable, which in turn facilitates the understanding of the optimizaiton quality for the specific plan shown in the map.

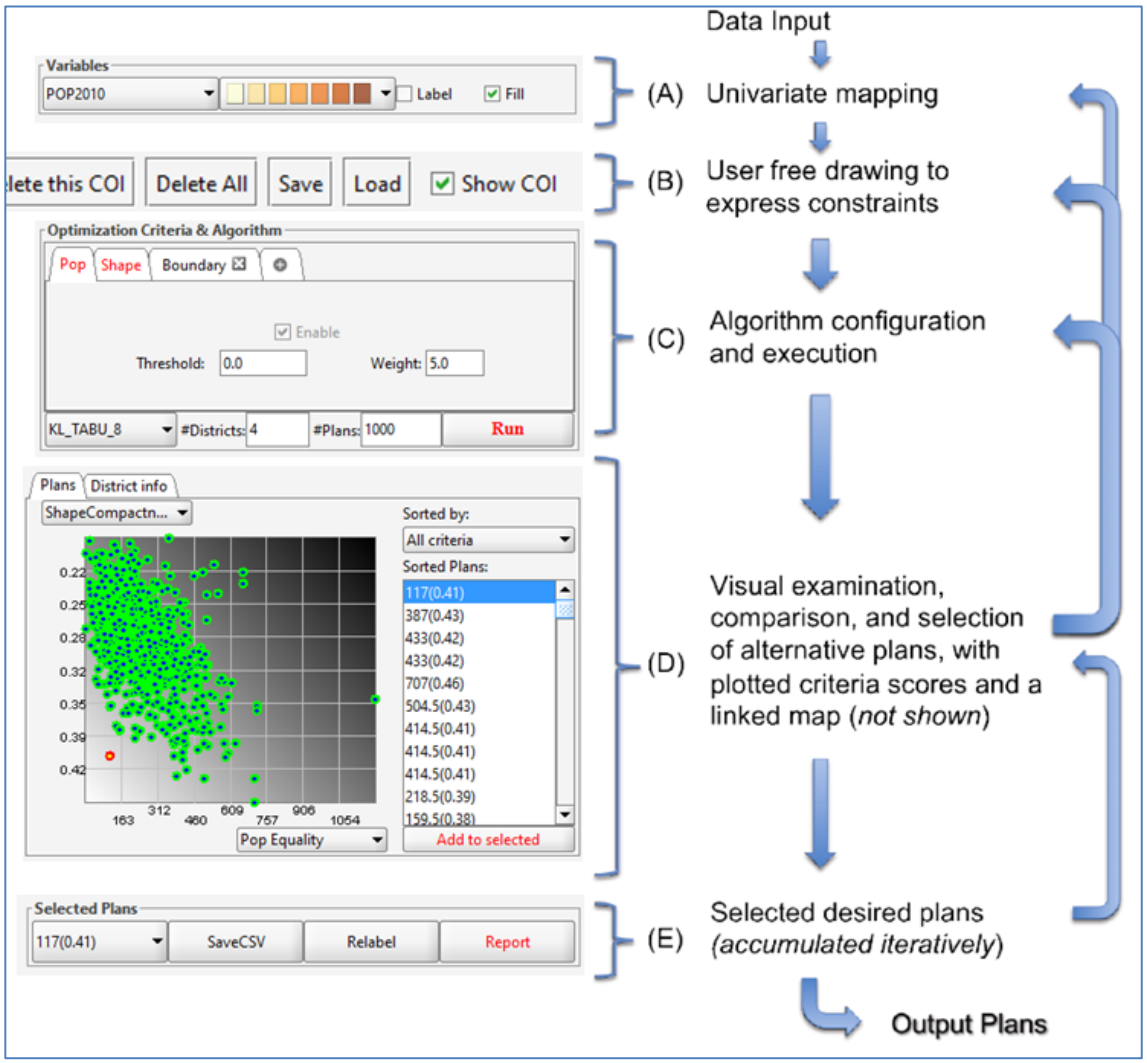

Figure 4.10 Visual interface to support an interactive and iterative optimization process

2) Community of Interest (COI)

A COI tool bar can be shown if the user wants to draw on the map to express constraints. It's a free drawing tool, which means the user can draw any type of polygon shapes. The user can click on the map to add a vertex, and double click to close the polygon. The unit objects intersected by the polygon will be considered as a COI, and thus won't be broken during the optimization process. The COI will be assigned a name and added to the COI dropdown list. The user-drawn COIs can be saved as a shapefile and loaded back later. Actually, the user can load any shapefile and choose some polygons from it to be COIs.

3) Algorithm configuration and execution

The user can enable the required criteria for redistricting and set the parameter for them such as the weight and the threshold (Figure 4.10(C)). More criteria can be added by clicking the "+" button on the criteria tab row. The user can choose the type of the criterion and input a unique name for it. A new tab for the added criterion will be shown. Unused criteria can also be removed. In this way, the user can easily define the required criteria for different redistricting problems.

The user can then configure the optimizaton algorithm, the number of districts, and the number of plans to be generated. By clicking the "run" button, the chosen optimization algorithm will be run with the configured criteria and constraints to generate the required number of redistricting plans.

4) Visual examination and comparison of alternative plans

After the redistricitng plans are generated, they are added to a plan list and shown on a scatter plot (Figure 4.10(D)). The default name of a plan is composed of the scores

of the two most common critera: popualtion equality and compactness. By default, the plan list is sorted by the combined score of all selected criteria (i.e the value of the objective function). The user can sort the plan list by the score of one criterion by using the “sorted by” dropdown list. The scatter plot shows the distribution of the scores of two criteria (population equality and compactness by default). The user can choose the criterion for each axis and compare the scores of different plans. The plan list and the scatter plot are linked, which means that clicking a plan on the list will highlight the plan on the scatter plot , and vice vesa. When a plan is clicked on the list or the scatter plot, the map will be updated to show its districts, and the report table in the report panel will be updated to show the attributes of the districts. The user can configure the attributes to be shown in the report table.

5) Managing plans

After examining and comparing the alternative plans, the user can add desired plans to the selected plan list (Figure 4.10(E)). The user can rename the selected plans. The labels of the districts in a selected plan can also be changed. The selected plans can be saved as a csv file, in which each column represents a plan. The saved csv file can be loaded back and added to the plan list. The current selected plan can also be saved as a shapefile of all districts, or several shapefiles each of which represents a district.

6) Locking districts

Sometimes an alternative plan is not good enough, but some of the districts are pretty good. In this case, the user can use the “lock” tool to lock the good districts and run the optimization algorithm again. The locked districts will be kept in the generated plans.

7) Editing plans

An “edit” tool is provided to change an alternative plan based on the user’s judgement. The user can drawn an polygon (like drawing COIs) to select some units, and a popup menu is then displayed to show a list of the districts that are neighboring these units. The user can choose the district these units will be moved to. If this change breaks the spatial contiguity, the user will be warned. The scatter plot, the plan list, and the report table will be updated after the change. If the user thinks the change is good, editing mode can be stopped and the change can be saved. The plan can be reverted to the previous state if the change is not good.

4.3 Computational solutions for handling large data volume

As explained in Chapter 3, the overall complexity of the optimization algorithm is $O(mkn\log n)$, where m is the number of criteria considered, k is the number of iterations during Tabu search, and n is the number of spatial units. To handle large datasets (e.g., n > 10,000), several strategies are developed by reducing k and/or n while not significantly sacrificing optimization quality.

4.3.1 Mega districts

In order to decrease the number of units in the redistricting optimization process, the whole area is divided into a small number of mega districts. The equal-population rule requires that the population of each mega district should be as close as possible to a whole number of ideal district population. Then, a redistricting process is done in each mega district. Let us use the largest state, California, as an example. California has 58 counties, 1081 census places/cities, around 25,000 VTDs, and about 500,000 blocks.

California needs 53 congressional districts. Keep in mind, as required by the redistricting rule, larger units should be used whenever possible. The user will first divide the state into a small number (of the user's choice) of mega districts at the county level. The equal-population rule requires that the population of each mega district should be as close as possible to a whole number of ideal district population. With the interactive process, a list of mega-district plans can be generated. If none of them sufficiently meets the equal population requirement, the user can break one or several counties into smaller units (such as VTDs) and optimize again. Once satisfied, the user then partitions each mega district separately. Since human's understanding of space is inherently hierarchical (Hirtle and Jonides 1985, Kuipers 2000), to divide a large state into many districts, it is more intuitive to take such a hierarchical process.

1) Mega district generation

The number of mega districts is determined by the defined max unit number of a mega district. The mega districts are generated using the same redistricting algorithm, but only the population and the shape compactness are considered. The target population for a mega district is a whole number of ideal district population so that the sum of the district numbers in mega districts is equal to the original required number of districts.

2) Plan generation

In each mega district, plans with a certain number of districts are generated using the redistricting algorithm, and the best plans are selected based on the measures. These best plans are then combined to form the final plans for the whole area. For example, if 2 best plans in each of 4 mega district are selected, $2^4 = 16$ final plans can generated by combining them.

Algorithm 4: Mega districts

Input:

n: the number of units in the whole area;

totalPop: the total population of the whole area;

r : the number of required districts;

k_i: the number of required districts in the i_{th} mega district;

Steps:

1. Divide the whole area into mega districts
   i. Set N_{max} to be the maximum number of units in a mega district;
   ii. Set the number of mega districts $= Min(Ceil(n / N_{max}), r/2)$, where Ceil() returns the smallest integer that is greater or equal to the input value. This makes sure the number of units in each mega district is less than N_{max} and each mega district has at least two districts;
   iii. Set $k_i = \frac{r}{m} (i = 1, 2, \dots, r)$;
   iv. Repeat the following until $\sum_{i=1}^{r} k_i = r$:
   k_i++;
   i++;
   v. SET the target population for the i_{th} mega district $targetPop_i = k_i * totalPop/r$;
   vi. Create *m* mega districts using the redistricting algorithm with the equal population and shape compactness criteria;
2. Run the redistricting algorithm in each mega district
   i. Split the original data set into each mega district;
   ii. Set the criteria for each mega district;
   iii. Create a certain number of sub-plans with k_i districts for the i_{th} mega district;
   iv. Select the top sub-plans based on the measure;
3. Combine the sub-plans in mega districts to form the final plans for the whole area
   i. Select one sub-plan in each mega district;
   ii. Combine these sub-plans to generate a final plan that covers the whole area.

4.3.2 Clustering

Clustering is a bottom-up process to decrease the number of units in the redistricting optimization process. The number of clusters is determined by a defined max population of a cluster. First, each unit is considered as a cluster. Then, a cluster is selected randomly, and its one neighbor is temporarily added to it. The measure of the new cluster is calculated and recorded, and the cluster is then reset. The neighbor that results in the best measure will be permanently added to the cluster. The process is repeated until there is no cluster to be merged under the max population constraint. These clusters are used instead of the original units in the redistricting process.

Algorithm 5: Clustering

Input:

n: the number of units in the whole area;

totalPop: the total population of the whole area;

C_{min}: the minimum number of clusters;

maxClusterPop : the maximum population of a cluster ($\mathrm{totalPop}/C_{\min}$);

Steps:

1. Each unit is considered as a cluster at the beginning;
2. Repeat the following steps until no clusters can be merged:
   i. Randomly select a cluster.
   ii. For each neighbor of the cluster:
      a. Temporarily add this neighbor to the cluster;
      b. If the population of the new cluster < maxClusterPop:
         Calculate the measure of the new cluster
   iii. Select the neighbor that results in the best measure and add it permanently to the cluster.

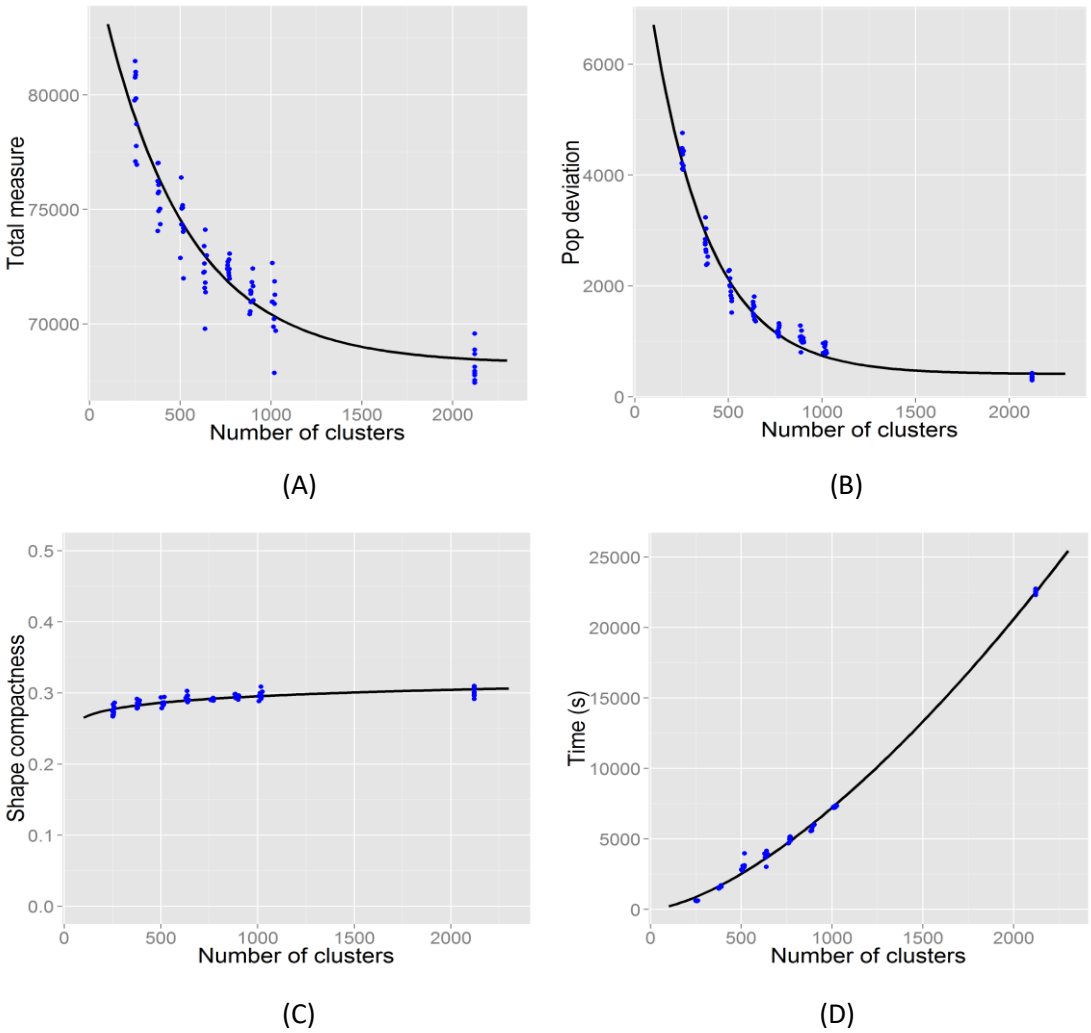

Figure 4.11 Results at different clustering levels with South Carolina data

Since it is slow to optimize both population equality and shape compactness with the South Carolina data used in Section 4.1.2, the clustering method can be used to improve the speed. 7 different clustering levels were used by setting the maximum population of a cluster, which was defined by a percentage of the total population of 2122 voting districts. The 7 different percentages used were 1/200, 1/300, 1/400, 1/500, 1/600, 1/700, and 1/800. At each clustering level, 10 sets of clusters were generated. For each set of clusters, 100 redistricting plans were derived while optimizing both population

equality and shape compactness. The results are shown in the Figure 4.11, and each point in the plots represents 100 redistricting plans. The running time decreases significantly when the number of clusters becomes small (Figure 4.11(D)), and the quality of the redistricting plans can still remain at an acceptable level. For example, at the 1000-cluster level, both the population equality and the shape compactness are still good enough, while the running time can be much shorter than that at the original unit level. In practice, the user can set the number of clusters based on the number of districts and the quality requirement.

4.3.3 Parallel and distributed computing

Mega districts and clustering are trying to decrease the number of units, while parallel and distributed computing is to do several things at the same time.

1) Multiple threads

The simplest implementation of parallel computing is that each thread generates a single plan at one time. The number of threads is determined by the hardware, and these threads are in a thread pool. When a thread finishes generating a plan, a new task is assigned to it. Each thread is relatively independent, except that some data objects are shared between threads.

When the plans in mega districts are generated, a thread can be created to run the redistricting process in a mega district. But different from multiple threads by plan, it's needed to wait for all the threads in all mega districts to finish and combine the top plans in each mega districts to form the final plans.

2) Hadoop

Hadoop is a reliable and scalable open-source framework for distributed computing across clusters of computers. The most important modules of Hadoop are its file system Hadoop Distributed File System (HDFS) and its data processing system Hadoop MapReduce. HDFS is a distribute file system that manage the data across clusters of computers. MapReduce is a programming model for parallel data processing. MapReduce is composed of two phases: a map phase and a reduce phase. The input and output of each phase are key-value pairs. The map phase takes the input as key-value pairs and generates zero or more key-value pairs. The reduce phase process the map output based on the map output keys and generates the final output.

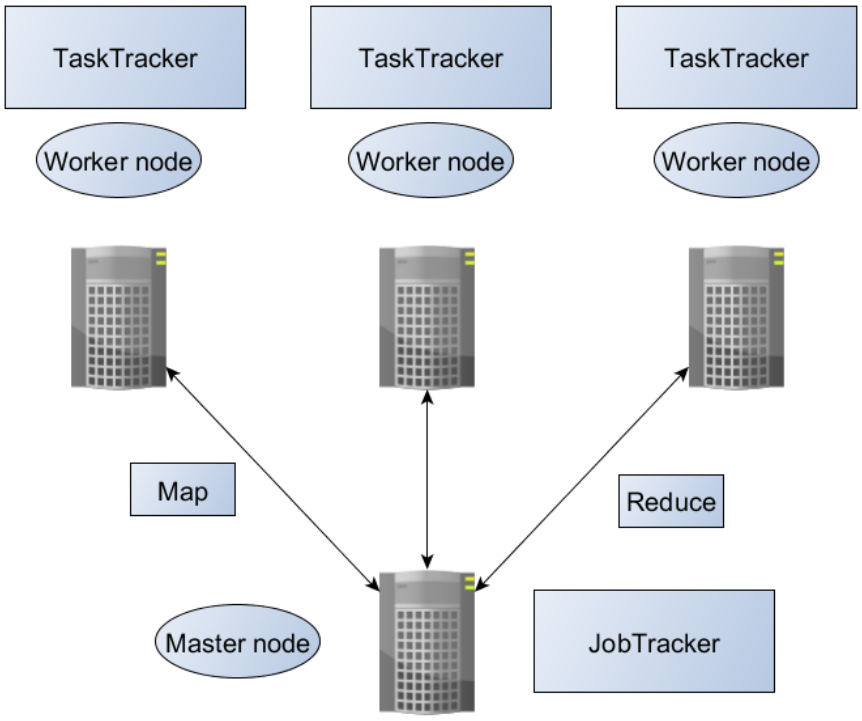

Figure 4.12 Hadoop

Hadoop assigns one computer in the cluster as the master node and other computers as the slave nodes. A JobTracker process is running on the master node, and a TaskTracker process in running on a slave node. The MapReduce client submits a job to the JobTracker, and the JobTracker schedules the map and reduce tasks on the TaskTracker processes across slave nodes.

In our Hadoop implementation, each line in the input data file is a JSON (JavaScript Object Notation) string containing the parameters for the redistricting algorithm. The input format class is set to be NLineInputFormat which takes N lines of the input as a split (N=1 by default). A Map function is implemented to generate a map task for each line. Each map task will run the redistricting algorithm based on the parameters in the line and output the plans in JSON as the value in the key-value pair. The key of the map output is the corresponding line number in the input data file. The JobTracker will then assign these map tasks to the TaskTrackers on the slave nodes. No reduce function is needed in our case. The outputs of the map tasks will be stored in a text file in which each line contains the line number in the input file and the JSON string of the plans.

3) Akka distributed system

A distributed system for running redistricting algorithms is implemented using the Akka framework. Akka is a toolkit and runtime for highly concurrent and distributed applications on the JVM. It's based on an actor model, in which actors send messages to interact with other actors. All interactions are asynchronous and thus suitable for a distributed environment.

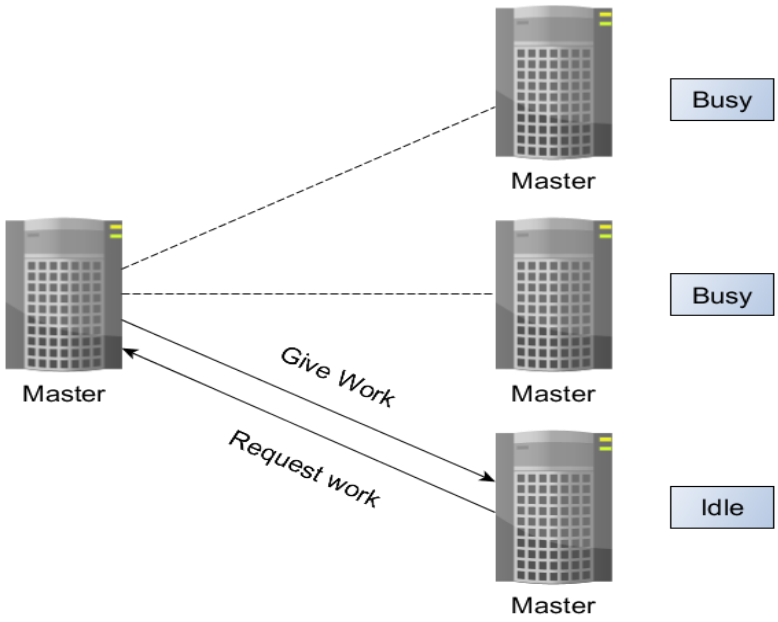

Figure 4.13 Akka distributed system

4.4 Conclusion

This chapter evaluates the new spatial optimization method with real-world case studies and compares it with existing methods. The results show that the new spatial optimization algorithm can significantly improve the computational efficiency and reliably achieve high optimization quality. It is flexible to consider different criteria and constraints for different districting problems. This chapter also presents the visual interface for user interaction to deal with subjective criteria, configure optimization criteria and parameters, and visually evaluate, explore, and manage optimization results. For large data sets, computational solutions are introduced to improve the computational efficiency.

CHAPTER 5
DISCOVER SPATIAL COMMUNITY STRUCTURE IN MOVEMENTS—AN EXTENTION OF THE OPTIMIZATION METHOD

This Chapter presents an extension of the optimization method to address a unique spatial optimization problem, which is to partition spatially embedded networks and discover spatial communities. Community structure detection is an important research topic in the analysis of complex networks, which is also essentially a combinatorial optimization problem, which is to find a partition of the network that optimizes an objective function.

In this Chapter, I focus on extending the optimization method introduced in Chapter 3 to detect spatial community structure from movements. Moreover, a series of new evaluations are carried out with synthetic datasets. This set of evaluations is different from the evaluation in Chapter 4 in that, the optimal solution is known with synthetic data and therefore it is possible to evaluate (1) whether the optimization method can discover the true pattern (global optima), (2) how different data characteristics may affect the performance of the method, and (3) whether the enforcement of spatial contiguity can help discover spatial structure, compared with non-spatial (general-purpose) optimization methods.

5.1 Introduction

Community structures in a complex network refer to the set of partitions where each respective community, or a group of nodes, has significantly more internal connections than connections to other groups. Community structures found in spatial networks, such as those present in networks capturing the movements of people or animals, are subject to contextual and environmental factors, physical constraints, and other spatial influences. Therefore detected communities in spatial networks often have geographic meanings such as urban boundaries, neighborhoods, and habitat territories. Existing literature has demonstrated extensive evidence that geographic mobility often exhibits strong spatial dependence and spatial community structures (Guo 2009a, Onnela et al. 2011, Comber, Brunsdon and Farmer 2012, Sun, Zheng and Hu 2012, Gao et al. 2013b, Chen, Xu and Xu 2015, Kallus et al. 2015).

Data on spatial movements have become increasingly available with the wide use of location-aware technologies such as GPS and smart phones. The analysis of movements is involved in a wide range of domains such as demography, migration, public health, urban study, transportation and biology. A movement data set consists of a set of moving objects, each having a sequence of sampled locations as the object moves across space, which forms a trajectory. The locations (points) in different trajectories are usually sampled independently and trajectory data can become very big such as billions of geotagged tweets, mobile phone records, floating vehicle locations, among others. Movements and trajectories can be analyzed to extract a variety of information such as points of interest or hot spots (Guo et al. 2012, Koylu and Guo 2013, Qi and Du 2013, Scholz and Lu 2014), flow patterns (Guo and Zhu 2014), and community structures

(Masser and Brown 1975, Guo 2009a, Sun et al. 2012, Kallus et al. 2015, Gao et al. 2013b).

Although numerous methodologies and applications have been proposed for detecting spatial communities, there is a lack of validations and evaluations of the correctness and robustness of respective methods. Movement data may be collected at different spatial and temporal resolutions and are subject to additional variation due to collection or instrumental errors and omissions. It is important to understand whether a method is robust in the presence of varying data characteristics and able to identify confirmed spatial community structures, by the application of validation steps.

Specifically, the research in this Chapter has two tasks. First, I adopt the modularity measure as the objective function and optimize the objective function with the new optimization method (Chapter 3) to discover community structure in a movement network. Second, I systematically evaluate and compare the results of the new method with commonly used methods in the literature, using synthetic datasets of known patterns and varying characteristics. This evaluation is different from the evaluation in Chapter 4 in that, the optimal solution is known with synthetic data and therefore it is possible to evaluate (1) whether the method can discover the true pattern (global optima), (2) how different data characteristics may affect the performance of the method, and (3) whether the enforcement of spatial contiguity can help discover spatial structures, compared with non-spatial (general-purpose) optimization methods.

Evaluation results reveal that general-purpose (non-spatial) methods are not robust in detecting spatial structures and may produce dramatically different outcomes for the same data with different characteristics, such as different spatial aggregations,

sampling rates, or noise levels. The new optimization method is significantly more stable and consistent. In addition to evaluations with synthetic datasets, a case study is also carried out to detect urban spatial structure with human movements, to demonstrate the application and effectiveness of the approach.

5.2 Related research

5.2.1 Movement Data Analysis

Based on the pattern types that a method can discover, movement analysis methods can be classified into a number of groups, including points of interest (hot spots or events) and their dynamic change over time (Guo et al. 2012, Koylu and Guo 2013, Qi and Du 2013, Scholz and Lu 2014), movement path and flow clusters (Guo and Zhu 2014, Yin and Shaw 2015), spatial community structures (Masser and Brown 1975, Guo 2009a, Sun et al. 2012, Kallus et al. 2015, Gao et al. 2013b), and spatial interaction models (Fotheringham 1983, Chen et al. 2016, Kang et al. 2015). A point of interest in movements can be defined in different ways depending on the application context. Density-based approaches are commonly used for detecting points of interest, such as the stacked space-time densities (Demsar et al. 2015), animal tracking data analysis with kernel density estimation (Downs and Horner 2012), heat maps of sports tracking data (Oksanen et al. 2015), and composite density maps for multivariate trajectories (Scheepens et al. 2011). Density-based approaches have also been developed for detecting movement path patterns and flow clusters such as flowing smoothing (Guo and Zhu 2014), graph bundling (Hurter, Ersoy and Telea 2012), and trajectory clustering (Zhu and Guo 2014, Rinzivillo et al. 2008). From a methodological perspective, Long et al.

(2013) classifies movement analysis methods into seven groups, including time geography, path descriptors, similarity indices, pattern and cluster methods, individual–group dynamics, spatial field methods, and spatial range methods.

In this research we focus on the detection of spatial community structure in movements, which can reveal unknown spatial/social/political boundaries, physical constraints and/or other spatial structures that govern mobility. Community structure detection has been extensively studied, from physics to complex networks, to biology, sociology, and spatial sciences (e.g., Girvan and Newman 2002, Guimera et al. 2005, Sun et al. 2012, Onnela et al. 2011, Rosvall and Bergstrom 2008). Community structure of a network exists if the network can be partitioned into groups of nodes such that the nodes in each group have strong connections with each other but much weaker connections to nodes outside their own communities. In a spatial network each node is a location and movements among locations are subject to the influence of geographic barrier, infrastructure, and other spatial constructs. The presence of spatial community structure in movement data was noted decades ago: "… there is a fundamental spatial organization in the pattern of movements and suggest that the discovery of this inherent system of migration regions is the most profitable avenue of approach to the present problem" (Ng 1969).

5.2.2 General-purpose Methods for Community Structure Detection

A general-purpose method for community structure detection normally involves two components: an *objective function* (or measure) that quantifies community strength and an *optimization method* that seeks to find the best partition of the network to

maximize the objective function. **Modularity** is one of the most commonly used objective functions (Newman and Girvan 2004, Newman 2006c, Newman 2006b), which measures community strength based on the difference between the observed and the expected connections within communities. For example, to partition an undirected network (graph) into two parts A and B, the best partition is the one that maximizes the total modularity within A and B. Let F be the total connections in a network, F_{AB} is the connection between A and B, F_{AA} and F_{BB} be the internal connections within A and B respectively, and $F_{A*} = F_{AA} + F_{AB}$ and $F_{B*} = F_{BB} + F_{AB}$ be the total connection incident on A and B. The modularity Q is then calculated as follows:

$Q(A,B)=F_{AB} – E_{AB}$, where $E_{AB}= F_{A*}F_{B*}/F$. (The modularity between A and B)

$Q(A,A)=F_{AA} – E_{AA}$, where $E_{AA}=F_{A*}F_{A*}/F$. (This is the modularity within A)

$Q(B,B)=F_{BB} – E_{BB}$, where $E_{BB}=F_{B*}F_{B*}/F$. (This is the modularity within B)

$Q = Q(A,A) + Q(B,B) = - Q(A,B)$. (The modularity for the partition into A and B)

By maximizing Q, i.e., $max_{A\subset G,B\subset G,A\cup B=G,A\cap B=\emptyset}(Q)$, we obtain the best two communities. This process is iterated to partition each subsequent community to build a hierarchy of communities.

Another objective function for community structure is the *edge ratio* (Cafieri, Hansen and Liberti 2010), which measures the strength of a community by the ratio between the number of connections within a region A (i.e., F_{AA}) to the number of connections that have only one end within A (i.e., F_{AB} for two-part partition). In dividing a graph into two components A and B, the smaller value of the two ratios for both A and B will be maximized, $max_{A\subset G,B\subset G,A\cup B=G,A\cap B=\emptyset}(\min(\frac{F_{AA}}{F_{AB}},\frac{F_{BB}}{F_{AB}}))$.

Different from the modularity and edge ratio, an information-theoretic approach (hereafter *Infomap*) uses a measure based on the Huffman code length of a random walker (Rosvall and Bergstrom 2008, Rosvall, Axelsson and Bergstrom 2009, Rosvall and Bergstrom 2011, De Domenico et al. 2015). The community structure of a network can be detected by minimizing the code length, which consists of the weighted entropy of the movement between communities and within communities. A more complete survey of various metrics (objective functions) for quantifying community structure is available in (Chakraborty et al. 2017).

Given an objective function, community detection is essentially an optimization problem, which is to find an optimal partition of the network that optimizes the objective function such as modularity, edge ratio, or the Huffman code length. However, as a combinatorial optimization problem, it is NP-hard and computationally challenging. Therefore, heuristic-based approaches are often used to search for near-optimal solutions, such as hierarchical clustering (Clauset, Newman and Moore 2004), matrix-based spectral clustering (Newman 2006c), iterative edge removal (Newman and Girvan 2004), multi-step greedy search (Schuetz and Caflisch 2008), multi-level merging (Blondel et al. 2008), greedy fine-tuning (Guo 2009b), and stochastic and recursive search (Pons and Latapy 2006, Rosvall and Bergstrom 2008).

5.2.3 Methods for spatial community structure detection

Current research on community structure in spatially embedded networks can be grouped into three types: (1) those that use a general-purpose (non-spatial) partitioning method to partition the network and then map the communities in geographic space to

examine spatial patterns; (2) those that modify the objective function of a general-purpose approach to integrate spatial factors (e.g., distance) or models (e.g., gravity model) and then partition the network by optimizing the modified objective function; or (3) those that enforce a spatial contiguity constraint in a general-purpose approach such that the discovered communities are spatially contiguous. In other words, the first type directly applies a non-spatial approach, the second type modifies the objective function, and the third type integrates spatial constraints in the optimization search.

Examples of the first type include using a general-purpose approach (e.g., maximizing modularity) to find community structures in land use (Comber et al. 2012), shipping network (Sun et al. 2012), social connections (Kallus et al. 2015), or urban population movements (Zhong et al. 2014). For example, Kallus et al. (2015) build a weighted spatial graph based on geo-located social networks and use a clustering method to maximize the modularity measure to find communities. Zhong et al. (2014) construct a weighted directed graph from urban travel records and use the Infomap method (Rosvall and Bergstrom 2008) to uncover urban community structure. While such direct applications of non-spatial methods can reveal interesting patterns, in this paper we will demonstrate its associated limitations and potential problems surrounding the detection of spatial community structures.

For the second type, Gao et al. (2013b) modify the modularity by using a gravity model to obtain the expected connections and defining "modularity" as a ratio comparing the actual to the expected connectivity within the network. Chen et al. (2015) propose a method that modifies the modularity by modifying the connection weight as the inverse of the geographic distance to the power of *n*. Integrating spatial models with non-spatial

measures is useful for specific purposes but introduces extra assumptions (e.g., distance decay) and more parameters, e.g., choices of different models and configuration of their parameters.

The third type of approach does not modify existing objective functions but adds a contiguity constraint to the optimization process and ensures that each community is spatially contiguous (Guo 2009b, Gao et al. 2013a, Guo, Liu and Jin 2010). Enforcing contiguity is common in spatial analysis practices such as regionalization (Assuncao et al. 2006, Guo 2008, Guo and Wang 2011), climate zoning (Fovell and Fovell 1993), image segmentation (Sharon et al. 2006), brain function analysis (Blumensath et al. 2013), and redistricting (Guo and Jin 2011b).

Although there have been extensive research efforts and numerous methods for community detection, there is a lack of research to evaluate the performance of different methods and validate the discovered communities (Yang and Leskovec 2015). This is partly due to the diversity of existing methods, the intractability of the optimization process, and the lack of benchmark datasets with known structures.

5.2.4 Methods for regionalization

Spatially constrained community structures are conceptually similar to outcomes produced by mainstream regionalization analysis. Regionalization (or regional partition) is a special form of spatial clustering that seeks to group spatial objects into spatially contiguous clusters while optimizing an objective function, which traditionally is based on univariate or multivariate similarities (Assuncao et al. 2006, Guo 2008, Guo and Wang 2011). Regionalization has been widely used in many application problems, such

as the delineation of climatic regions and eco-regions. The approach presented in this paper can be considered a new type of regionalization, which is different from traditional regionalization in terms of input (which is a spatial complex network instead of multivariate spatial data), method (which is contiguity constrained partitioning and optimization instead of multivariate clustering), and output (where each region is a densely-connected community instead of a homogenous area).

5.3 Detecting spatial community structure with optimization

First, movement data are transformed to a node-link graph (complex network), in which each node is a location and the connection between two nodes is the weighted total of moving objects that have visited both nodes.

Second, the graph is partitioned with the new optimization method to discover spatial community structure. A spatial community is a geographically contiguous region with more internal movements than external movements (i.e., movements to the outside). Depending on the application context, a spatial community may represent a habitat territory for animals or an urban functional region within which people conduct most of their daily activities.

5.3.1 Trajectory data representation and graph construction

A movement data set consists of a set of moving objects, each of which has a sequence of sampled locations as the object moves across space, which forms the trajectory of the object. The locations (points) in different trajectories are usually sampled independently. Let T =(Jones et al.), i=1..n, be a set of n trajectories, each trajectory T_i = {<s_{ij}, $j_{ij,}$ w_{ij}>} is a sequence of points (or areas), and each point has a location s_{ij}, a time

stamp t_{ij}, and an optional weight value w_{ij} to indicate the importance of the location in that trajectory. By default the weight for each location is one if trajectory points are collected at a regular time interval. If data points are sampled at irregular time intervals, the time duration at each location can be used as the weight. Let $S = \{s_{ij}\}$, $|S| = m$, be the set of unique locations from all trajectories. A graph G of these m locations is constructed. For each trajectory T_i, every unique pair of locations s_{ij} and s_{ik} ($j \neq k$) on the trajectory adds a weight $w_{ij} * w_{ik}$ to the edge $< s_{ij}, s_{ik} >$ in G. Figure 5.1 illustrates the graph construction process with three simple trajectories. If there are no identical locations in trajectories, which is the case in Figure 5.1, then G consists of n disconnected sub-graphs, each being a single trajectory. Colors in Figure 5.1 are used for illustration only—G does not distinguish edges from different trajectories.

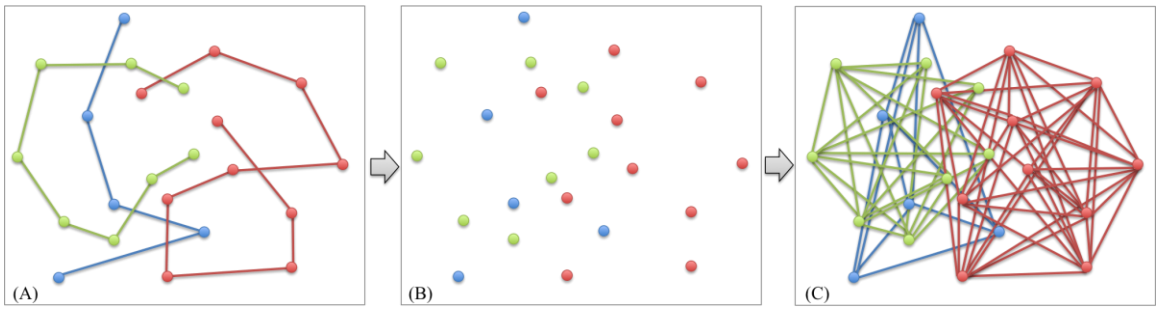

Figure 5.1 An illustration of graph construction from trajectories: (A) three trajectories, (B) locations from all trajectories, and (C) the weighted graph G constructed from trajectories.

For big datasets with a large number of location points, an initial spatial clustering of the points can be performed to reduce unnecessary data detail and also help improve computational efficiency in the optimization step. For example, in a dataset of Taxi trips, there may be thousands of drop-off or pick-up points (GPS locations) within 20 meters to

a subway station, in which case it is not necessary to treat each point as a unique location. Grouping them into clusters would not substantially affect the data analysis outcomes when the study area is large, such as a city (Zhu and Guo 2017, Guo et al. 2012). A simple k-means clustering (with k being the number of clusters) of the locations based on spatial coordinates can serve this purpose. The original graph G is then aggregated with the k clusters as nodes. In the experiments in Section 5.2, we use different k values to examine the effect of spatial aggregation on the analysis results, for both our new approach and other approaches.

To build the spatial contiguity relationship among points, Thiessen polygons (i.e., Voronoi diagrams) are constructed for the m locations in $S = \{s_{ij}\}$ so that each location point is enclosed by a polygon. Two location points are considered spatially contiguous if their polygons share a border. If points are grouped into spatial clusters, each cluster is also a polygon, which is the union of its member polygons (points). The contiguity matrix is aggregated accordingly.

5.3.2 Detection of spatial communities (districts)

With the modularity measure (Newman and Girvan 2004) as the objective function for quantifying community structure, the optimization method in Chapter 3 can be applied to group nodes (spatial locations) into geographically contiguous regions (or communities) by optimizing the objective function. However, there is one major difference in this process since the number of communities (regions) is unknown. Therefore, a hierarchical approach is taken—the optimization method will first construct two communities (regions), and then find the best partition of the discovered regions so

far to produce one more region. This process will be repeated to build a hierarchy of communities (regions).

Specifically, given an input graph *G*, three steps are followed to partition the graph and discover hierarchical spatial communities. *First*, an initial partition P = (A, B) is constructed, where $A \cup B = G$ and $A \cap B = \emptyset$, using a contiguity-constrained hierarchical clustering method as explained in (Guo 2009a). *Second*, this initial partition is optimized with the new Tabu-search algorithm as explained in Chapter 3, which iteratively evaluates all possible moves (i.e., moving points) between A and B without breaking the spatial contiguity of each and find the best move to change the partition (A, B) to a new partition (A*, B*), where A* and B* are the new communities, $A^* \cup B^* = G$ and $A^* \cap B^* = \emptyset$. This process is iterated until no further improvement can be found after a long sequence of non-improving moves. *Third*, for each of the resulted communities, repeat the above steps (initial partition and Tabu optimization) to find the best community that should be cut (and its best cut), which will result one more community in the hierarchy. This repeats until no more cut can be found (e.g., the modularity cannot be improved).

Due to its ability to escape local optimum, the Tabu search has more optimization power than existing methods such as fast approximation methods (Blondel et al. 2008), hierarchical clustering (Gao et al. 2013b), and greedy local searches (Guo 2009a). Given an initial solution, the Tabu search not only guarantees to find the local optimum but also attempts to go beyond. It achieves this by using a Tabu list to keep the most recent moves, which are not allowed to move again—as such the search can be forced out of the neighborhood of the local optimum and lead to better solutions. The complexity of the algorithm is $O(c^2)$, where $c = m$ if no spatial clustering or c = the number of clusters. The

optimization process takes about 5 minutes on a desktop computer with a 3.2Hz CPU for $c = 5000$ and less than 1 second for $c < 500$. The contiguity constraint helps to achieve such a computational efficiency since the number of possible moves is reduced to that of moves that maintain contiguity.

5.4 Evaluation with synthetic data

In this section I select a number of representative methods to be evaluated and compared, including *Modularity* (general purpose), *Infomap* (general purpose), and *modularity with contiguity constraint* (the Tabu optimization method in Chapter 3 with a contiguity constraint). For the convenience of reference, the new spatial optimization method is called **STOCS** (Spatial Tabu Optimization for Community Structure) in this Chapter. The purpose of the evaluation with synthetic data is to examine how robust and effective the three methods (i.e., Modularity, Infomap, and STOCS) are in detecting valid spatial structures from data with different levels of spatial aggregation, data sampling, and data noise.

5.4.1 Synthetic data

I generate a series of synthetic trajectory datasets within a rectangular study area R with the following steps. First, the rectangular area R is divided into four arbitrary-shaped regions: A, B, C, and D (Figure 5.2). Second, six clusters of trajectories are generated, each cluster has 20 trajectories, and each trajectory (a moving object) has 50 sampled location points. The spatial distribution of points on a trajectory depends on which cluster it is in (Table 5.1).

In this research the order of points on a trajectory is not important, I do not simulate the actual sequence and only sample locations based on specific requirements as explained below. For cluster 1, the movements are predominantly (90% of the points) in region A, with 10% random moves (points) anywhere in R. Similarly, trajectories in cluster 2 are mainly in region B, cluster 3 mainly in region C, and cluster 4 mainly in region D. No matter which cluster it is in, each trajectory has 10% of points (which can be considered noise) randomly placed in R. For cluster 5, each trajectory has statistically equal presence in both A and B (each having 45%), plus 10% random moves in R. Cluster 6 is similar to cluster 5 except that its dominant regions are C and D. Note that each trajectory has 10% of noise or random moves. Third, add a set of 20 random trajectories, each of which moves freely in R without any spatial pattern. Figure 5.2 shows the four regions and a total of 7000 location points from all 140 trajectories. Table 5.1 shows the configuration of the six clusters and the random set. I also generated a second dataset with 20% of noise in each trajectory (Table 5.2).

A weighted graph *G* and a spatial contiguity matrix are constructed with the 7000 trajectory points, as explained in Section 5.3.1. Each location has a default weight of one. Since the locations are unique, graph *G* has 140 disconnected sub-graphs, each being a complete graph of the points on a trajectory (as illustrated in Figure 5.1). To examine the effect of spatial aggregation on each method, a spatial *k*-means clustering is performed to aggregate the 7000 points into 3000, 1000, and 300 spatial clusters respectively. Graph *G* and its spatial contiguity matrix are then aggregated accordingly. As such, there are six different input graphs that respectively have 3000 nodes, 1000 nodes, and 300 nodes, for the data with 10% noise and the data with 20% noise.

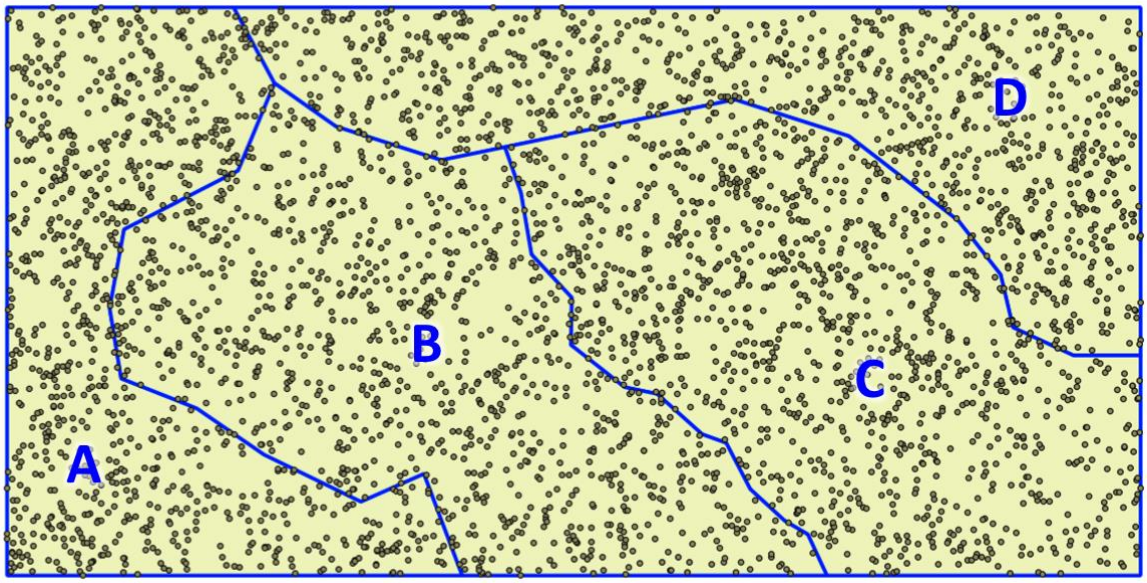

Figure 5.2 Synthetic data for experiments.

Table 5.1 Synthetic data of trajectories with 10% noise or random moves.

Cluster ID	Number of Trajectories	For each trajectory in a cluster					
		Portion in A	Portion in B	Portion in C	Portion in D	Random Points	Total
1	20	90%				10%	100%
2	20		90%			10%	100%
3	20			90%		10%	100%
4	20				90%	10%	100%
5	20	45%	45%			10%	100%
6	20			45%	45%	10%	100%
Random Set	20	Points in each trajectory are random chosen within the entire rectangular area					100%

Table 5.2 Synthetic data of trajectories with 20% noise or random moves.

Cluster ID	Number of Trajectories	For each trajectory in a cluster					
		Portion in A	Portion in B	Portion in C	Portion in D	Random Points	Total
1	20	80%				20%	100%
2	20		80%			20%	100%
3	20			80%		20%	100%
4	20				80%	20%	100%
5	20	40%	40%			20%	100%
6	20			40%	40%	20%	100%
Random Set	20	Points in each trajectory are random chosen within the entire rectangular area					100%

5.4.2 Evaluation results

Each of the six input graphs is partitioned with the three selected methods: Infomap, modularity (without contiguity constraint), and STOCS (i.e., spatial Tabu optimization with modularity as the objective function and with a spatial contiguity constraint). Figure 5.3 shows the results for the data of 10% noise at different spatial aggregation levels. Each discovered community is represented by a unique and randomly assigned color. At the lowest level with 3000 clusters (each cluster has two or three points on average), the communities found by both the Infomap and the original modularity methods cannot reveal a clear spatial structure, with communities intertwined in space. As explained earlier, when trajectories do not share locations, the graph has isolated communities formed by trajectories and the general-purpose methods cannot effectively identify spatial structure.

STOCS, on the other hand, can clearly identify the four community regions. More aggressive spatial aggregation (1000 clusters) substantially helps the Infomap and original modularity methods to recognize the embedded spatial structure, although there are still spatially scattered pieces in each community. For the new approach, the result remains consistent with the true pattern and high-level aggregation leads to smoother region boundaries. Further aggregation to 300 clusters allows the two general-purpose methods to better identify true spatial communities. However, too much aggregation causes another problem, e.g., Infomap can only identify three regions (which is correct but misses the internal structure within the larger region). In summary, the results shown in Figure 5.3 indicate that the two general approaches (without contiguity constraint) can partially discover the embedded spatial structure with a suitable spatial aggregation level

but are very sensitive to the change of spatial aggregations. Since the appropriate aggregation level is unknown beforehand and may vary from pattern to pattern, it is not reliable to use general-purpose approaches to detect spatial structure. With a contiguity constraint, the new approach is robust for different aggregations.

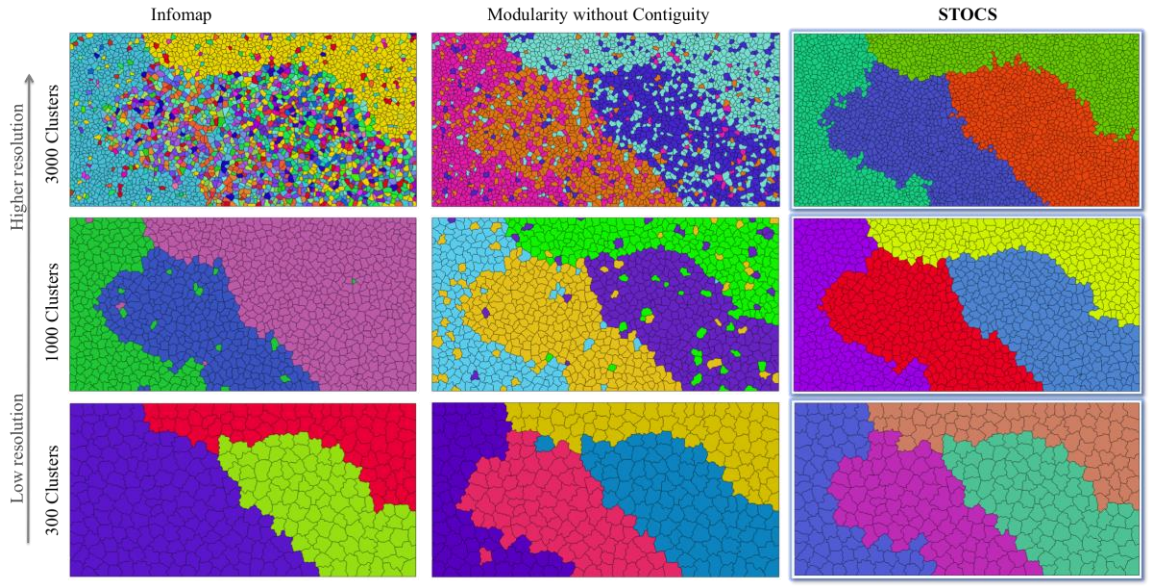

Figure 5.3 Experimental results for data with 10% noise or random moves (Table 5.1). The new approach can discover the top two regions (AB and CD) at the two-region level (which is not shown here) and the four regions at the next hierarchy (as shown in the three maps of the last column). Each community is represented with a randomly assigned unique color.

Figure 5.4 shows results for the 20% noise data, which can help understand the effect of more noise or random moves. With more noise and thus fewer spatial signals, Infomap cannot find any separated communities at the 1000- and 300-cluster level since locations are more connected due to random connections. The general modularity is better than Infomap but also degrades with increasing noise. The new approach remains consistent across aggregations with increased noise. The contiguity constraint can effectively filter out non-spatial noise and focus on spatial structures. Note that, although

results of the new approach in Figure 5.3 and Figure 5.4 show four communities, it also correctly identifies the two larger communities at the higher level.

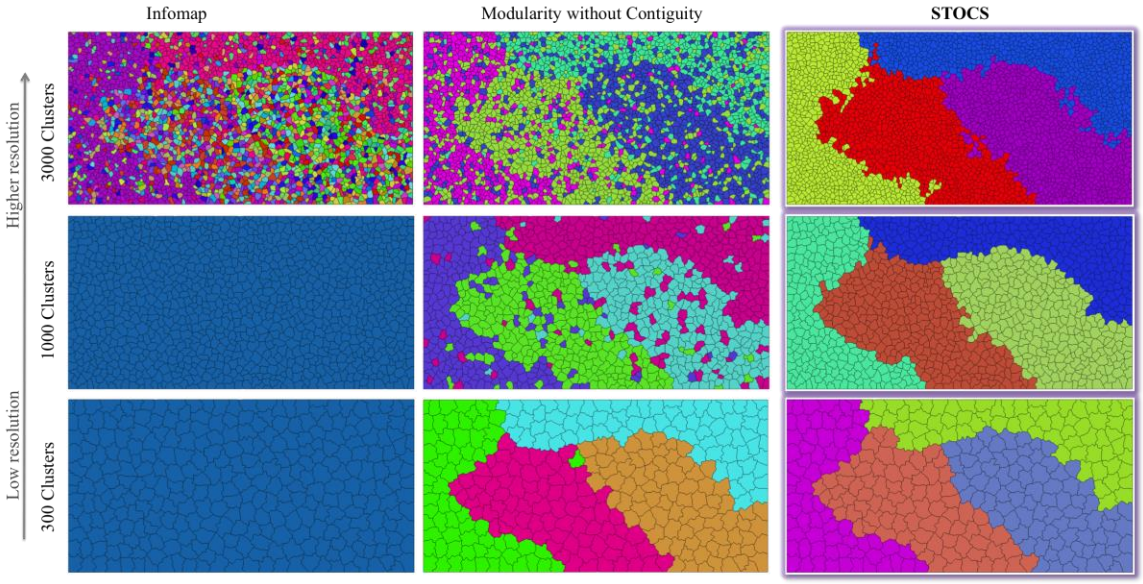

Figure 5.4 Experiment result for the data with 20% noise or random moves (Table 5.2).

To quantitatively compare the quality of the community structures detected by these methods, I use the normalized mutual information (NMI) measure to evaluate how well the discovered communities match the ground-truth communities (which are the four spatial communities as shown in Figure 5.2). The NMI measure is widely used for comparing community detection methods (Mahmood et al. 2017, Emmons et al. 2016, Chakraborty et al. 2017), which is defined as follows. Let N = 7000 be the total number of spatial points, $\Omega = \{w_1, w_2, \ldots, w_K\}$ be the set of detected spatial communities (each of which is a set of spatial points), and $C = \{c_1, c_2, \ldots, c_J\}$ be the ground-truth communities. $NMI(\Omega, C) = I(\Omega, C)/([H(\Omega) + H(C)]/2)$, where I is the mutual information between Ω and C: $I(\Omega, C) = \sum_k \sum_j \frac{|w_k \cap c_j|}{N} log \frac{N|w_k \cap c_j|}{|w_k||c_j|}$ and H is the entropy:

$H(\Omega) = -\sum_k \frac{|w_k|}{N} log \frac{|w_k|}{N}$, $H(\mathrm{C}) = -\sum_j \frac{|c_j|}{N} log \frac{|c_j|}{N}$. NMI values range between 0 and 1, with 1 representing a perfect match. Note that we do not expect any result to be exactly the same as the ground truth communities (Figure 5.2) since the experiment data sets are stochastic realizations and may deviate slightly from the designed boundaries.

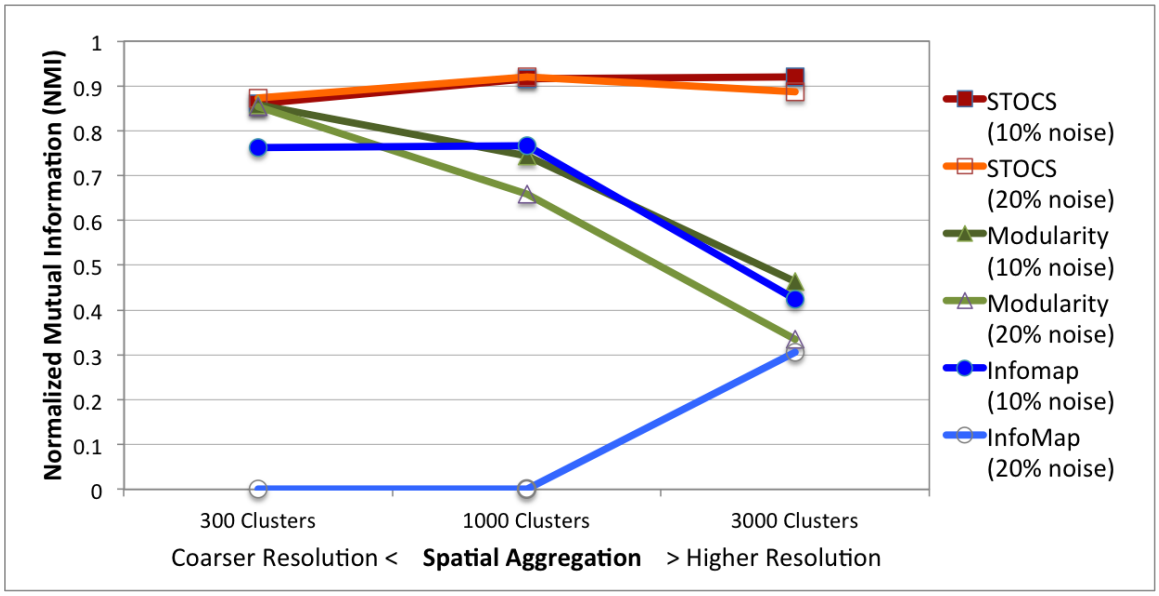

Figure 5.5 Normalized mutual information (NMI) values of each result. Higher NMI values represent better matching with the ground-truth, with 1 being a perfect match.

Figure 5.5 shows the NMI values for the results in Figure 5.3 and Figure 5.4. For the 10%-noise datasets, the three methods are comparable at the 300-cluster level but the performance of both Infomap and Modularity (without contiguity) degrades quickly with more clusters (thus higher resolution). On the opposite, STOCS (our method) achieves even better performance with more clusters (i.e., less spatial aggregation). For 20%-noise datasets, STOCS maintains the same performance across different levels of aggregation while the performance of the other two methods drops substantially with more noise.

In addition to the effects of spatial aggregation and data noise as presented above, I also examined how data sampling may affect the performance of those methods. This can help us understand how the sampling rate of trajectory points may affect the performance of different methods. Using the data of 10% noise and with 1000 clusters (i.e., the center row in Figure 5.3), I randomly draw 60% sample locations from each trajectory. In other words, 40% of location points in each trajectory are discarded. With this sampled data, the results of the three methods are shown in Figure 5.6. With data points reduced, the results of the two general-purpose methods deteriorate significantly, while STOCS still reveals a clear spatial structure, although boundaries are less smooth.

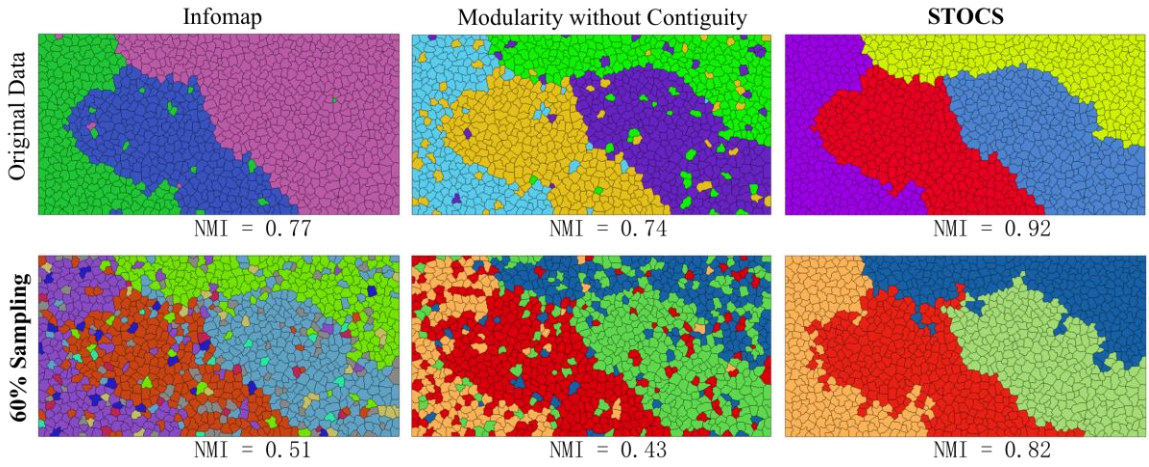

Figure 5.6 The top row shows the community detection results of the original synthetic data with 1000 clusters (as in Figure 5.3). The bottom row shows the results after randomly removing 40% points. The NMI score is shown for each map.

To summarize, with seven synthetic datasets, I systematically examined the effects of three factors, *spatial aggregation*, *data noise* and *data sampling*, on the performance of three selected methods in detecting the true spatial community structure embedded in movements. These three factors are common in real-world movement data

and thus pose critical challenges for movement analysis. The experimental results indicate that a direct application of non-spatial general-purpose methods is sensitive to spatial aggregation, noise, or sampling, which may produce spurious patterns (e.g., spatially scattered pieces of a community) or miss important structures. Modified objective functions that integrate spatial models or distance often introduce additional assumptions (e.g., distance decay) and more parameters (e.g., choices of different models and configuration of their parameters), which may bias towards certain patterns. Our approach (STOCS) with a contiguity constraint is robust and can reliably detect spatial community structure without imposing assumptions regarding the region shape or distance effect.

5.5 Case study with urban population movements

To further evaluate the applicability of the approach for handling much larger data and different trajectory data types, a case study is carried out with a big dataset of anonymized mobile phone records (CDR—Call Detail Records), which has over 2.5 million mobile phone users (devices) for one day (from midnight to midnight) in Shanghai, China. A data record represents a user activity with the mobile service, e.g., calling, messaging, or surfing internet, with start time, duration, and cellular station identifier (and location). There are about 5000 cellular stations, which on average are 300-500 meters apart from each other, closer in downtown area and farther in suburban area. A user on average has more than 400 records a day, of varying temporal distributions. With the records for each user, I estimate the time that the user stayed at each station and the stay time is used as the weight of the station for this user.

Now a graph *G* can be built with the 5000 cellular stations as nodes (thus the clustering step is not needed), following the procedure in Section 5.3. The connection weight between station A and station B contributed by a user will be his/her stay time at A multiplies his/her stay time at B. Note that the stay time of a person for a cellular station is the sum of time between two consecutive records of this person at the station within 30 minutes. In other words, a time gap (i.e., no data) of more than 30 minutes will not be counted as stay time for any station. Repeating the process for all users will generate a pairwise station-to-station network *G* with accumulated weights for each connection, contributed by all users. This process ensures that (1) the importance of each location is considered (by stay time); and (2) every user can contribute regardless of their sampling rate.

Given *G*, our contiguity-constrained modularity method discovered 11 urban communities (regions) when the modularity reaches the maximum. To check the meaning and validate, the discovered spatial communities are overlaid with the boundaries of 18 administrative districts of Shanghai (Figure 5.7). It is interesting to see that the discovered spatial community boundaries largely follow the administrative boundaries, particularly for the suburban area (for example, communities 4, 8, 10, 11). Since the spatial communities are formed by the Thiessen polygons of cellular stations, they would not be able to exactly follow the administrative boundaries. The two airports, both located on district borders, play an important role in shaping the communities.

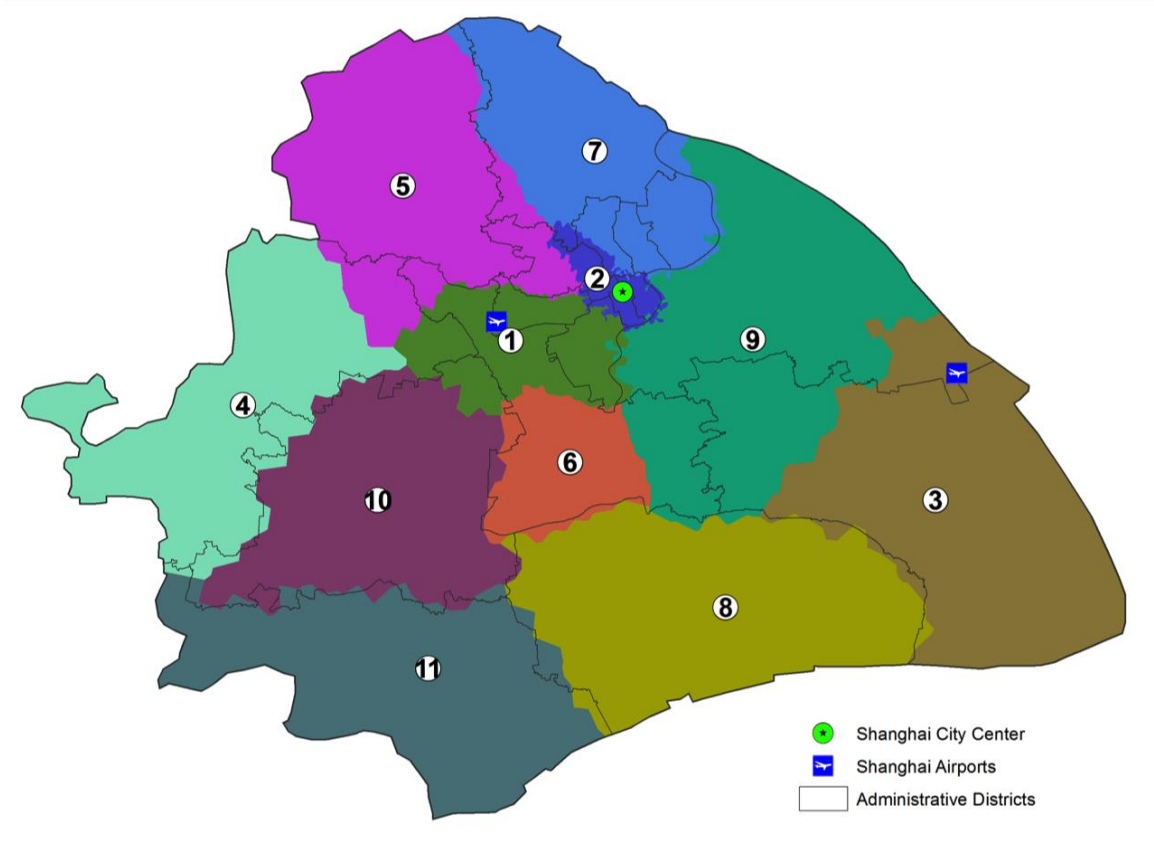

Figure 5.7 Eleven discovered spatial communities (colored polygons) from the mobile phone data in Shanghai, compared with administrative district boundaries (gray line)

To further verify the regions, I calculate the percentage of time that each user spent in each region. For this part the focus is on the 1,674,754 users that have more than 2 hours of total activity time (i.e., the stay time at all visited stations) to ensure reliable percentage calculation. There are more than 65% of people (i.e., over one million users) who spent more than 95% of their time in a single spatial community (region). Ninety percent (90%) of users have a primary community, in which he/she spent more than 60% of the time. In other words, if a user is assigned to his/her primary community (where he or she spent >60% of his/her time), we can effectively group 90% of the 1,674,754 users into 11 clusters, one for each of the discovered communities.

5.6 Conclusion

This research presents an extension of the spatial optimization method (Chapter 3) to detect spatial community structure from complex networks established by movements. It transforms trajectory data to a spatial complex network and then partitions the network by optimizing a chosen objective function (modularity) with the new contiguity constrained spatial Tabu optimization method (STOCS). I designed a series of synthetic datasets with known patterns to systematically evaluate and compare STOCS with selected methods. Evaluation results show that general-purpose non-spatial methods highly depend on appropriate spatial aggregation to discover the true spatial community structure. With different levels of aggregation, data noise or data sampling, non-spatial methods may produce dramatically different results for the same data. STOCS is significantly more robust and consistent. A case study is carried out with urban population movements to demonstrate the application and effectiveness of the approach. The discovered spatial communities not only help understand mobility patterns but can also enable the extraction of trajectory features and further movement.

The new approach can filter out noise and detect spatial community structure in movements. The definition of contiguity is flexible and can be customized according to application needs. For example, the contiguity constraint does not have to be spatial—it can be a constraint of other types, such as social relations, depending on applications. The contiguity constraint is optional but critically important for detecting spatial structures, as shown in the experiments. By turning on or off the contiguity constraint, we can switch between a general-purpose method and a spatially constrained method for the same

objective function. The contiguity constraint also significantly improves the computational efficiency as it reduces the search space. On the other hand, the Tabu-based search is powerful and can achieve a high optimization power under the contiguity constraint. If two remote and non-contiguous areas are strongly connected, it will be more suitable to represent the pattern as a flow pattern rather than merging the two areas into a "community". From the other perspective, one should be cautious when seeing spatially dispersed communities from a non-spatial method, which may be spurious patterns from random moves, as shown in our experiments. New statistical methods are needed to test the significance of such movement patterns.

CHAPTER 6
DISCUSSION AND FUTURE WORK

This research is to develop a computation-assisted and user-centered approach to address practical redistricting problems and other spatial optimization problems. A spatial optimization method and a system (*iRedistrict*) have been developed to deal with practical challenges in different redistricting problems. The major contribution of this research is that the developed methods and system are efficient, flexible, and powerful so that it is practically useful for users of different skill levels.

First, the developed spatial optimization method can optimize multiple criteria under multiple constraints, and be able to construct high-quality optimization solutions. Second, the developed redistricting system integrates the developed optimization method with intuitive user interfaces and user interaction strategies to leverage human judgments and computing power for tackling complex redistricting tasks. Both the spatial optimization methods and the system can be easily adapted to meet requirements for different redistricting problems. Case studies in political redistricting, school redistricting, and community structure detection have been carried out to show how the developed spatial optimization methods and system work with real-world problems in a wide range of domains.

In future, the spatial optimization methods can be further improved to be more efficient and powerful, and the redistricting system can be user-friendly for novice users.

Combinatorial optimization problems are inherently challenging. The current method and system reported in this dissertation still struggles with very large datasets. With future work, the spatial optimization method can be implemented in a truly parallel and distributed environment. Currently the method can use multiple threads, which does improve the efficiency to generate multiple plans or multiple mega districts at the same time. However, the optimization process for generating a single plan or a single mega district still uses a single computing unit and thus is not parallelizable. It can be improved by evaluating candidate moves or switches in a parallel manner. Since the number of the candidate moves at each step can be very large, the parallel architecture of GPU may be helpful in this process. The methods can also be integrated with the Spark distributed memory-based computing framework.

Another very important direction for future work is to be able to consider hierarchical spatial units simultaneously in the optimization process. Currently the generated redistricting plan has to be based on units at the same level. For some redistricting problems, units at different levels can be combined in the final redistricting plan. The spatial optimization methods should be able to switch between different unit levels at different locations when it is appropriate.

The visual interface of the redistricting system can be improved to be more intuitive. Redistricting by nature is a complex process. Innovative user interface and interaction strategies are needed, preferably web-based. Expanding to different domains, we will need to incorporate more criteria to address new redistricting problems. For example, it might be possible to extend the optimization method to solve routing

optimization problems, which will bring in a different set of optimization criteria and constraints.

Another remaining challenge yet to be solved is the parameter configuration for the Tabu heuristic, which involves a number of parameters such as the length of the Tabu list and the number of maximum iteration without improvement. There is no theoretical guidance on how these parameters should be set in relation to specific data characteristics. As the case study for South Carolina congressional redistricting shows, the performance of the optimization method depends on parameter settings. Currently, the parameters are set with empirical knowledge.

REFERENCES

Altman, M. (1997) Is automation the answer: The computational complexity of automated redistricting. *Rutgers Computer and Law Technology Journal,* 23**,** 81-141.

Altman, M. & M. P. McDonald (2009) BARD: Better Automated Redistricting. *Journal of Statistical Software*.

Assis, L. S. d., P. M. Franca & F. L. Usberti (2014) A redistricting problem applied to meter reading in power distribution networks. *Computers & Operations Research,* 41**,** 65-75.

Assuncao, R. M., M. C. Neves, G. Camara & C. Da Costa Freitas (2006) Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees. *International Journal of Geographical Information Science,* 20**,** 797-811.

Bacao, F., V. Lobo & M. Painho (2005) Applying genetic algorithms to zone design. *Soft Computing,* 9**,** 341-348.

Battiti, R. & A. A. Bertossi (1999) Greedy, prohibition, and reactive heuristics for graph partitioning. *Computers, IEEE Transactions on,* 48**,** 361-385.

Bennett, A. R. 2010. Home health care logistics planning. United States -- Georgia: Georgia Institute of Technology.

Bergey, P., C. Ragsdale & M. Hoskote (2003) A simulated annealing genetic algorithm for the electrical power districting problem. *Annals of Operations Research,* 121**,** 33-55.

Blondel, V. D., J. L. Guillaume, R. Lambiotte & E. Lefebvre (2008) Fast unfolding of communities in large networks. *Journal of Statistical Mechanics-Theory and Experiment***,** P10008.

Blum, C. & A. Roli (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys,* 35**,** 268-308.

Blumensath, T., S. Jbabdi, M. F. Glasser, D. C. Van Essen, K. Ugurbil, T. E. J. Behrens & S. M. Smith (2013) Spatially constrained hierarchical parcellation of the brain with resting-state fMRI. *Neuroimage,* 76**,** 313-324.

Bozkaya, B. 1999. Political districting: A tabu search algorithm and geographical interfaces. 193 p. Canada: University of Alberta (Canada).

Bozkaya, B., E. Erkut & G. Laporte (2003) A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research,* 144**,** 12-26.

Browdy, M. (1990) Simulated annealing: An improved computer model for political redistricting. *Yale Law & Policy Review,* 8**,** 163.

Cafieri, S., P. Hansen & L. Liberti (2010) Edge ratio and community structure in networks. *Physical Review E,* 81**,** 026105.

Caro, F., T. Shirabe, M. Guignard & A. Weintraub (2004) School redistricting: embedding GIS tools with integer programming. *Journal of the Operational Research Society,* 55**,** 836-849.

Castelli, M., R. Henriques & L. Vanneschi (2015) A geometric semantic genetic programming system for the electoral redistricting problem. *Neurocomputing,* 154**,** 200-207.

Chakraborty, T., A. Dalmia, A. Mukherjee & N. Ganguly (2017) Metrics for Community Analysis: A Survey. *ACM Computing Surveys,* 50**,** Article 54, https://doi.org/10.1145/3091106.

Chance, C. (1965) Political studies: Number 2-representation and reappointment. *Dept. of Political Science*.

Chen, B. Y., H. Yuan, Q. Q. Li, S. L. Shaw, W. H. K. Lam & X. L. Chen (2016) Spatiotemporal data model for network time geographic analysis in the era of big data. *International Journal of Geographical Information Science,* 30**,** 1041-1071.

Chen, Y., J. Xu & M. Z. Xu (2015) Finding community structure in spatially constrained complex networks. *International Journal of Geographical Information Science,* 29**,** 889-911.

Chou, C., S. Kimbrough, J. Sullivan-Fedock, C. J. Woodard & F. Murphy. 2012. *Using Interactive Evolutionary Computation (IEC) with Validated Surrogate Fitness Functions for Redistricting*.

Clauset, A., M. E. Newman & C. Moore (2004) Finding community structure in very large networks. *Phys Rev E Stat Nonlin Soft Matter Phys,* 70**,** 066111.

Comber, A. J., C. F. Brunsdon & C. J. Q. Farmer (2012) Community detection in spatial networks: Inferring land use from a planar graph of land cover objects. *International Journal of Applied Earth Observation and Geoinformation,* 18**,** 274-282.

D'Amico, S. J., S. J. Wang, R. Batta & C. M. Rump (2002) A simulated annealing approach to police district design. *Computers & Operations Research,* 29**,** 667-684.

De Domenico, M., A. Lancichinetti, A. Arenas & M. Rosvall (2015) Identifying Modular Flows on Multilayer Networks Reveals Highly Overlapping Organization in Interconnected Systems. *Phys. Rev. X,* 5**,** 11027-11027.

Demsar, U., K. Buchin, E. E. van Loon & J. Shamoun-Baranes (2015) Stacked space-time densities: a geovisualisation approach to explore dynamics of space use over time. *Geoinformatica,* 19**,** 85-115.

Di Cortona, P., C. Manzi, A. Pennisi, F. Ricca & B. Simeone. 1999. *Evaluation and optimization of electoral systems*. Society for Industrial Mathematics.

Dorigo, M., G. D. Caro & L. M. Gambardella (1999) Ant algorithms for discrete optimization. *Artificial life,* 5**,** 137-172.

Downs, J. A. & M. W. Horner (2012) Analysing infrequently sampled animal tracking data by incorporating generalized movement trajectories with kernel density estimation. *Computers Environment and Urban Systems,* 36**,** 302-310.

Emmons, S., S. Kobourov, M. Gallant & K. Borner (2016) Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. *Plos One,* 11.

Forman, S. 2002. Congressional Redistricting Using a TSP-based Genetic Algorithm. In *Genetic and Evolutionary Computation Conference*, 1262. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Forrest, E. (1964) Apportionment by computer. *American behavioral scientist,* 8**,** 23.

Fotheringham, A. S. (1983) A New Set of Spatial-Interaction Models: The Theory of Competing Destinations. *Environment and Planning A,* 15**,** 15-36.

Fovell, R. G. & M. Y. C. Fovell (1993) Climate Zones of the Conterminous United-States Defined Using Cluster-Analysis. *Journal of Climate,* 6**,** 2103-2135.

Gabow, H. N. 2000a. Path-based depth-first search for strong and bi-connected components. Boulder, CO.: Department of Computer Science, University of Colorado at Boulder.

--- (2000b) Path-based depth-first search for strong and biconnected components. *Information Processing Letters,* 74**,** 107-114.

Gao, P., J. A. Kupfer, D. S. Guo & T. L. Lei (2013a) Identifying functionally connected habitat compartments with a novel regionalization technique. *Landscape Ecology,* 28**,** 1949-1959.

Gao, S., Y. Liu, Y. L. Wang & X. J. Ma (2013b) Discovering Spatial Interaction Communities from Mobile Phone Data. *Transactions in GIS,* 17**,** 463-481.

Garfinkel, R. & G. Nemhauser (1970) Optimal political districting by implicit enumeration techniques. *Management Science,* 16**,** 495-508.

Gearhart, B. & J. Liittschwager (1969) Legislative districting by computer. *Behavioral Science,* 14**,** 404-417.

Girvan, M. & M. E. Newman (2002) Community structure in social and biological networks. *Proc Natl Acad Sci U S A,* 99**,** 7821-6.

Glover, F. (1990) Tabu search, Part I1. *ORSA Journal on computing,* 2**,** 4-32.

González-Ramírez, R. G., N. R. Smith, R. G. Askin, P. A. Miranda & J. M. Sánchez (2011) A hybrid metaheuristic approach to optimize the districting design of a parcel company. *Journal of Applied Research and Technology,* 9**,** 19-35.

Guimera, R., S. Mossa, A. Turtschi & L. A. N. Amaral (2005) The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences of the United States of America,* 102**,** 7794-7799.

Guo, D. (2008) Regionalization with dynamically constrained agglomerative clustering and partitioning (REDCAP). *International Journal of Geographical Information Science,* 22**,** 801-823.

--- (2009a) Flow Mapping and Multivariate Visualization of Large Spatial Interaction Data. *IEEE Transactions on Visualization and Computer Graphics (TVCG: Proc. of InfoVis'09),* 15**,** 1041-1048.

Guo, D. & H. Jin (2011a) iRedistrict: Geovisual analytics for redistricting optimization. *Journal of Visual Languages & Computing*.

Guo, D., S. Liu & H. Jin (2010) A Graph-based Approach to Vehicle Trajectory Analysis. *Journal of Location Based Service,* 4**,** 183 - 199.

Guo, D. S. (2009b) Flow Mapping and Multivariate Visualization of Large Spatial Interaction Data. *IEEE Transactions on Visualization and Computer Graphics,* 15**,** 1041-1048.

Guo, D. S. & H. Jin (2011b) iRedistrict: Geovisual analytics for redistricting optimization. *Journal of Visual Languages and Computing,* 22**,** 279-289.

Guo, D. S. & H. Wang (2011) Automatic Region Building for Spatial Analysis. *Transactions in GIS,* 15**,** 29-45.

Guo, D. S. & X. Zhu (2014) Origin-Destination Flow Data Smoothing and Mapping. *IEEE Transactions on Visualization and Computer Graphics,* 20**,** 2043-2052.

Guo, D. S., X. Zhu, H. Jin, P. Gao & C. Andris (2012) Discovering Spatial Patterns in Origin-Destination Mobility Data. *Transactions in GIS,* 16**,** 411-429.

Hess, S., J. Weaver, H. Siegfeldt, J. Whelan & P. Zitlau (1965) Nonpartisan political redistricting by computer. *Operations Research,* 13**,** 998-1006.

Hirtle, S. C. & J. Jonides (1985) Evidence of hierarchies in cognitive maps. *Memory & Cognition,* 13**,** 208-217.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.

Hu, F., S. Yang & W. Xu (2014) A non-dominated sorting genetic algorithm for the location and districting planning of earthquake shelters. *International Journal of Geographical Information Science,* 28**,** 1482-1501.

Huntley, C. L. 1996. Simulated annealing for multiobjective, hierarchical service districting problems. 115 p. United States -- Virginia: University of Virginia.

Hurter, C., O. Ersoy & A. Telea (2012) Graph Bundling by Kernel Density Estimation. *Computer Graphics Forum,* 31**,** 865-874.

Jones, C. B., R. S. Purves, P. D. Clough & H. Joho (2008) Modelling vague places with knowledge from the Web. *International Journal of Geographical Information Science,* 22**,** 1045-1065.

Joshi, D. 2011. Polygonal spatial clustering. United States -- Nebraska: The University of Nebraska - Lincoln.

Kalcsics, J., S. Nickel & M. Schröder (2005) Towards a unified territorial design approach-Applications, algorithms and GIS integration. *Top,* 13**,** 1-56.

Kallus, Z., N. Barankai, J. Szule & G. Vattay (2015) Spatial Fingerprints of Community Structure in Human Interaction Network for an Extensive Set of Large-Scale Regions. *Plos One,* 10.

Kang, C. G., Y. Liu, D. S. Guo & K. Qin (2015) A Generalized Radiation Model for Human Mobility: Spatial Scale, Searching Direction and Trip Constraint. *Plos One,* 10.

Kernighan, B. W. & S. Lin (1970) An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal,* 49**,** 291-307.

Kim, M. J. 2011. Optimization Approaches to Political Redistricting Problems. United States -- Ohio: The Ohio State University.

Kirpatrick, S., C. Gelatt & M. Vecchi (1983) Optimization by simulated annealing. *Science,* 220**,** 671-680.

Ko, J., E. Nazarian, Y. Nam & Y. Guo (2015) Integrated redistricting, location-allocation and service sharing with intra-district service transfer to reduce demand overload and its disparity. *Computers, Environment and Urban Systems,* 54**,** 132-143.

Koylu, C. & D. S. Guo (2013) Smoothing locational measures in spatial interaction networks. *Computers Environment and Urban Systems,* 41**,** 12-25.

Kuipers, B. (2000) The Spatial Semantic Hierarchy. *Artificial Intelligence,* 119**,** 191-233.

Levitt, J. & B. Foster (2008) A Citizen's Guide to Redistricting. 113. http://www.brennancenter.org/content/resource/a_citizens_guide_to_redistricting/ (last accessed Feb. 28, 2010).

Liu, Y. Y., W. K. T. Cho & S. Wang (2016) PEAR: a massively parallel evolutionary computation approach for political redistricting optimization and analysis. *Swarm and Evolutionary Computation,* 30**,** 78-92.

Long, J. A. & T. A. Nelson (2013) A review of quantitative methods for movement data. *International Journal of Geographical Information Science,* 27**,** 292-318.

Lourenco, H., O. Martin & T. Stützle (2003) Iterated local search. *Handbook of metaheuristics***,** 320-353.

Macmillan, W. (2001) Redistricting in a GIS environment: An optimisation algorithm using switching-points. *Journal of geographical systems,* 3**,** 167-180.

Mahmood, A., M. Small, S. A. Al-Maadeed & N. Rajpoot (2017) Using Geodesic Space Density Gradients for Network Community Detection. *IEEE Transactions on Knowledge and Data Engineering,* 29**,** 921-935.

Masser, I. & P. J. B. Brown (1975) Hierarchical aggregation procedures for interaction data. *Environment and Planning A,* 7**,** 509-523.

Mehrotra, A., E. Johnson & G. Nemhauser (1998a) An optimization based heuristic for political districting. *Management Science,* 44**,** 1100-1114.

Mehrotra, A., E. L. Johnson & G. L. Nemhauser (1998b) An optimization based heuristic for political districting. *Management Science,* 44**,** 1100-1114.

Nagel, S. (1965) Simplified bipartisan computer redistricting. *Stanford Law Review,* 17**,** 863-899.

Nemhauser, G. L. & L. A. Wolsey. 1999. *Integer and combinatorial optimization*. Wiley New York.

Newman, M. E. (2006a) Modularity and community structure in networks. *Proceedings of the National Academy of Sciences,* 103**,** 8577-8582.

Newman, M. E. & M. Girvan (2004) Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys,* 69**,** 026113.

Newman, M. E. J. (2006b) Finding community structure in networks using the eigenvectors of matrices. *Physical Review E,* 74.

--- (2006c) Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America,* 103**,** 8577-8582.

Ng, R. C. Y. (1969) Recent Internal Population Movement in Thailand. *Annals of the Association of American Geographers,* 59**,** 710-730.

Novaes, A. G. N., J. E. S. de Curst, A. C. L. da Silva & J. C. Souza (2009) Solving continuous location-districting problems with Voronoi diagrams. *Computers & Operations Research,* 36**,** 40-59.

Oksanen, J., C. Bergman, J. Sainio & J. Westerholm (2015) Methods for deriving and calibrating privacy-preserving heat maps from mobile sports tracking application data. *Journal of Transport Geography,* 48**,** 135-144.

Onnela, J. P., S. Arbesman, M. C. Gonzalez, A. L. Barabasi & N. A. Christakis (2011) Geographic Constraints on Social Network Groups. *Plos One,* 6.

Openshaw, S. (1977) Optimal zoning systems for spatial interaction models. *Environment and Planning A,* 9**,** 169-184.

Papadimitriou, C. H. & K. Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Dover publications.

Polsby, D. D. & R. D. Popper (1991) Partisan Gerrymandering: Harms and a New Solution. *Heartland Policy Study,* 34.

Pons, P. & M. Latapy (2006) Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications,* 10**,** 191–218.

Puppe, C. & A. Tasnadi (2008) A computational approach to unbiased districting. *Mathematical and Computer Modelling,* 48**,** 1455-1460.

Qi, F. & F. Du (2013) Trajectory Data Analyses for Pedestrian Space-time Activity Study. *Jove-Journal of Visualized Experiments*.

Ricca, F., A. Scozzari & B. Simeone (2008) Weighted Voronoi region algorithms for political districting. *Mathematical and Computer Modelling,* 48**,** 1468-1477.

Ricca, F. & B. Simeone (2008) Local search algorithms for political districting. *European Journal of Operational Research,* 189**,** 1409-1426.

Rincón-García, E. A., M. A. Gutiérrez-Andrade, S. G. de-los-Cobos-Silva, P. Lara-Velázquez, A. S. Ponsich & R. A. Mora-Gutiérrez (2013) A Multiobjective Algorithm for Redistricting. *Journal of Applied Research and Technology,* 11**,** 324-330.

Rinzivillo, S., D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko & G. Andrienko (2008) Visually driven analysis of movement data by progressive clustering. *Information Visualization,* 7**,** 225-239.

Rosvall, M., D. Axelsson & C. T. Bergstrom (2009) The map equation. *The European Physical Journal Special Topics,* 178**,** 13-23.

Rosvall, M. & C. T. Bergstrom (2008) Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America,* 105**,** 1118-23.

--- (2011) Multilevel Compression of Random Walks on Networks Reveals Hierarchical Organization in Large Integrated Systems. *PLOS ONE,* 6**,** e18209-e18209.

Sammons, R. 1978. A simplistic approach to the redistricting problem. In *Spatial representation and spatial interaction,* eds. I. Masser & P. Brown, 71-94. Boston: Martinus Nijhoff Social Sciences Division.

Scheepens, R., N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko & J. J. van Wijk (2011) Composite Density Maps for Multivariate Trajectories. *IEEE Transactions on Visualization and Computer Graphics,* 17**,** 2518-2527.

Scholz, R. W. & Y. M. Lu (2014) Detection of dynamic activity patterns at a collective level from large-volume trajectory data. *International Journal of Geographical Information Science,* 28**,** 946-963.

Schuetz, P. & A. Caflisch (2008) Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E,* 77.

Segura-Ramiro, J., R. Rıos-Mercado, A. M. Álvarez-Socarrás & K. de Alba Romenus. 2007. A location-allocation heuristic for a territory design problem in a beverage distribution firm. In *International Conference on Industrial Engineering*, 428-434. Cancun, Mexico.

Sharon, E., M. Galun, D. Sharon, R. Basri & A. Brandt (2006) Hierarchy and adaptivity in segmenting visual scenes. *Nature,* 442**,** 810-813.

Sun, Z., J. F. Zheng & H. T. Hu (2012) Finding Community Structure in Spatial Maritime Shipping Networks. *International Journal of Modern Physics C,* 23.

Tarjan, R. (1972) Depth-first search and linear graph algorithms. *SIAM journal on computing,* 1**,** 146-160.

Vanneschi, L., R. Henriques & M. Castelli (2017) Multi-objective genetic algorithm with variable neighbourhood search for the electoral redistricting problem. *Swarm and Evolutionary Computation,* 36**,** 37-51.

Vickrey, W. (1961) On the prevention of gerrymandering. *Political Science Quarterly,* 76**,** 105-110.

Williams, J. C. (1995) POLITICAL REDISTRICTING: A REVIEW. *Papers in Regional Science,* 74**,** 13-40.

Xiao, N. 2003. Geographical optimization using evolutionary algorithms. 119 p. United States -- Iowa: The University of Iowa.

--- (2008) A Unified Conceptual Framework for Geographical Optimization Using Evolutionary Algorithms. *Annals of the Association of American Geographers,* 98**,** 795-817.

Yamada, T. (2009) A mini-max spanning forest approach to the political districting problem. *International Journal of Systems Science,* 40**,** 471-477.

Yang, J. & J. Leskovec (2015) Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems,* 42**,** 181-213.

Yin, L. & S. L. Shaw (2015) Exploring space-time paths in physical and social closeness spaces: a space-time GIS approach. *International Journal of Geographical Information Science,* 29**,** 742-761.

Zhang, Y. & D. E. Brown (2013) Police patrol districting method and simulation evaluation using agent-based model & GIS. *Security Informatics,* 2**,** 7-7.

Zhong, C., S. M. Arisona, X. F. Huang, M. Batty & G. Schmitt (2014) Detecting the dynamics of urban structure through spatial network analysis. *International Journal of Geographical Information Science,* 28**,** 2178-2199.

Zhu, X. & D. Guo (2017) Urban event detection with big data of taxi OD trips: A time series decomposition approach. *Transactions in GIS,* 21**,** 560–574.

Zhu, X. & D. S. Guo (2014) Mapping Large Spatial Flow Data with Hierarchical Clustering. *Transactions in GIS,* 18**,** 421-435.