OPTIMIZATION APPROACHES FOR POLITICAL DISTRICTING AND GRAPH PARTITIONING

BY

RAHUL SWAMY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Doctoral Committee:

Teaching Assistant Professor Douglas M. King, Chair and Co-Director of Research
Professor Sheldon H. Jacobson, Co-Director of Research
Professor Carolyn L. Beck
Assistant Professor Jugal Garg

Abstract

Graph partitioning (GP) is the problem of dividing a graph into smaller subgraphs with desired properties. GP is valuable in a variety of domains such as social, transportation and power networks. A notable application of GP is in political districting in the U.S., where congressional and state legislative districts are redrawn every ten years. Existing studies that model political districting as a GP have explored exact, heuristic, and game theoretical techniques. However, these studies have three key drawbacks: (i) they disregard political fairness considerations that are increasingly valued in legal requirements, (ii) the methods are computationally intractable to large input sizes, and (iii) they do not capture the numerous and complex needs of a practical districting process. This dissertation provides optimization formulations for fair political districting and heuristic frameworks that scale for large input sizes. This dissertation consists of four parts. The first part addresses the issue of fairness in political districting by providing Mixed Integer Programming (MIP) formulations that optimize fundamental fairness such as efficiency gap, partisan asymmetry, and competitiveness. Three bi-criteria problems are solved using a proposed multilevel algorithm, which generates approximate-Pareto optimal solutions, illustrating the trade-off between compactness and each of the partisan fairness metrics. The second part provides a case study on districting in Arizona, capturing all the legal criteria in Arizona's Constitution. This multi-stage local-search powered framework demonstrates how optimization algorithms can generate viable solutions to practical districting. The third part provides heuristic strategies for two sequential game protocols, namely the *bisection* and *I-cut-you-freeze* protocols, where the drawing decision in each round is modeled as an MIP. A computational investigation for real-world congressional districting in 18 states indicates that neither protocol is fairer than the other, although both provide an improvement to the status quo. Beyond political districting, the fourth part introduces a variant of GP with high connectivity requirements, called the *Highly Connected Graph Partitioning* (HCGP) problem. This problem is valuable in applications that require cohesion and fault-tolerance within their parts, such as community detection in social networks and resiliency-focused partitioning of power networks. This work provides an Integer Programming model for HCGP and solution methods to solve instances derived from a diverse set of real-world graphs. Overall, the models and methods presented in this dissertation serve as effective tools to capture fairness in political districting and resiliency in GP.

I dedicate this thesis to my parents, Kalai Selvi and Vairavel,
my sister Sindhu,
and my grandparents Meenatchi and Ponnusamy.

Acknowledgements

I begin by acknowledging the immense support I received throughout my Ph.D. journey; it takes a village. I am profoundly grateful to my advisors, Dr. Sheldon Jacobson and Dr. Douglas King, whose guidance has been instrumental in helping me navigate the intricate landscape of academia. Their mentorship has been invaluable in shaping the course of my research. Their commitment to my academic growth has left an enduring mark on my journey, and I am truly grateful for this.

I thank my committee members, Dr. Carolyn Beck and Dr. Jugal Garg, for generously dedicating their time and expertise to serve on my committee. Their mentorship has played a significant role in making this dissertation possible. I also extend my thanks to the esteemed faculty and dedicated staff at UIUC. The faculty who guided me through my coursework have provided me with a robust foundational knowledge upon which I have prepared this dissertation. I extend my thanks to the dedicated staff at the Department of Industrial and Enterprise Systems Engineering for their support in ensuring a smooth academic experience.

I thank my friends and peers at the University of Illinois at Urbana-Champaign (UIUC) for their support and presence. Special thanks go to Sai Mali Ananthanarayanan and Svetlana Borodina for being there during significant moments and providing me with the support and motivation to persevere. While the list is not extensive and not in any specific order, I extend my thanks to my peers at UIUC, including Timothy Murray, Setareh Taki, Kiera Dobbs, Ian Ludden, Arash Khatibi, Hee Youn Kwon, Wenda Zhang, and Sagnik Das. Through every challenge faced and milestone achieved, I cherish their role in shaping the person I have become.

Finally, I extend my heartfelt thanks to my family, whose constant love and support have been instrumental in my journey. Their presence in my life is a source of comfort and inspiration for which I am deeply grateful.

This work, through Dr. Jacobson, was partly supported by the Air Force Office of Scientific Research (FA9550-19-1-0106). Any opinions, findings, and conclusions or recommendations expressed in this dissertation are mine, and do not necessarily reflect the views of the United States Government, or the Air Force Office of Scientific Research.

Table of Contents

List of Abbreviations

Below is the list of all abbreviations used in this dissertation arranged in alphabetical order.

AIRC	Arizona Independent Redistricting Commission,
CD118	Congressional District map for the 118^{th} Congress,
CN	Continuous Non-Geometric Setting,
COI	Community of Interest,
CPU	Central Processing Unit,
D	Democratic Party,
DG	Discrete Geometric Setting,
DP	Districting Problem,
EG	Efficiency Gap,
GP	Graph Partitioning,
HCB(T)	Hybrid Census Block (Tract),
HCBG	Hybrid Census Block Group,
HCGP	Highly Connected Graph Partitioning,
ICYF	I-Cut-You-Freeze Protocol,
IP	Integer Program,
MCMC	Markov Chain Monte-Carlo,
MI(L)P	Mixed Integer (Linear) Program,
MK Growth	Multi-Kernel Growth,
ML(P)	Minmax Maximal Matching (Problem),
MM(P)	Minmax Maximum Matching (Problem),
NP	Nondeterministic Polynomial,
PA	Partisan Asymmetry,
PDP	Political Districting Problem,
PFDP	Politically Fair Districting Problem,
R	Republican Party,
U.S.	United States,
VRA	Voting Rights Act.

Chapter 1

Introduction

Once every ten years, state and federal institutions in the United States undertake the task of districting (or redistricting) a state into political districts, which are commonly known as district maps or district plans. The precise way these districts are drawn affects how the voters are spatially divided, influencing electoral representation and the political landscape of a legislature. Given its potential impact, *gerrymandering*, the act of intentionally manipulating the map-drawing process to achieve particular political objectives that stakeholders may consider "unfair" is commonly observed. Incorporating fairness in political districting is essential to maintain equity among the various stakeholders, such as political parties, candidates, voters, and independent commissions.

The far-reaching societal implications of political districting have spurred research studies across various disciplines, including political sciences, operations research, mathematics, and geography. Existing research falls into two broad categories: (i) generation algorithms focused on constructing district maps and (ii) tools designed to assess the potential gerrymandering of a given map. A map-drawing authority, such as an independent redistricting commission, can use a map-generation algorithm to create legally valid maps. Among these algorithms, optimization methods are well-suited to optimize fairness criteria. However, existing optimization models and methods predominantly prioritize non-political criteria, which may not align with notions of political equity. For example, the classical graph partitioning-based Political Districting Problem (PDP) optimizes the compactness of districts (measuring the geographical shapes of the districts) with constraints on population balance and contiguity. However, solely optimizing compactness could inadvertently "pack" urban voters into fewer districts, diminishing their electoral representation. Moreover, on the assessment of gerrymandering front, political science and mathematics literature has established numerous mathematical and computational tools (e.g., Markov chain Monte Carlo) to assess whether a district map is an outlier in terms of a political fairness metric. Popular metrics such as the efficiency gap and partisan asymmetry capture fairness concerning the political parties, while competitiveness focuses on the voters. While these metrics and tools are valuable in legal conflicts arising after enacting a map, there is potential for incorporating a fairness-focused optimization algorithm beforehand to reduce the likelihood of legal disputes. Thus, there is a need to develop optimization models and methods that explicitly incorporate fairness objectives.

Despite the potential benefits of using an optimization algorithm for districting, significant challenges must be overcome for practical adoption. The first challenge is the complexity of modeling the districting process. Practical districting often requires the consideration of a wide range of criteria. For example, Arizona's Constitution requires numerous criteria such as contiguity, compactness, preservation of political subdivisions,

population balance, majority-minority districts, communities of interest, and political competitiveness. However, existing optimization methods have primarily focused on a subset of three criteria: population balance, contiguity, and compactness. Capturing all the legal criteria in a state's constitution has proven to be a significant challenge, partly due to the subjectivity of these criteria and the lack of transparent data. For example, it is common to require that districts preserve "communities of interest," but it is often unclear which communities the map-drawers have chosen to preserve by the end of the process. The second challenge is computational intractability. Many optimization methods cannot scale to the large input sizes needed for practical districting. For example, it takes more than three days for an exact method to solve an instance of PDP with nearly a thousand census tracts (Validi et al., 2022). However, the number of census blocks in practical districting can range from thousands to nearly one million, as in the case of Texas. The third challenge is the alignment with a practical districting procedure. For advanced technology to assist with practical adoption, it must be transparent and capable of integrating with established practices. For example, Arizona uses a multi-stage process where a district map undergoes a series of modifications while incorporating public feedback. However, mathematical optimization methods such as a Mixed Integer Program (MIP) can be opaque to non-experts in the field, which could impede their implementation in practice. Hence, an optimization-based map-generation algorithm must address these challenges.

In addition to a one-shot map-generation algorithm, an alternative method for promoting partisan fairness is to frame districting as a strategic game between political parties. The literature proposes sequential games, where two players (i.e., political parties) take turns to construct a district map. Each player's overall objective is to maximize the number of districts they win (i.e., in which they have a simple majority). Two recently introduced sequential games are the *I-cut-you-freeze* (ICYF) protocol (Pegden et al., 2017) and the *bisection* protocol (Ludden et al., 2023). The ICYF protocol involves two players alternating between *drawing* a district plan in the remaining region (originating from the whole state) and *freezing* one of those districts. The bisection protocol involves two players taking alternating turns to bisect the remaining region, starting from the entire state, until a district plan is created. Existing work has derived closed-form expressions for the players' optimal strategies, albeit for a simplistic setting that treats voters as continuous entities and disregards their spatial distribution within a state. Nevertheless, computing optimal strategies is computationally intractable for a more realistic setting that incorporates the discrete (e.g., in voting precincts or census tracts) and geometric distribution of voters. Despite this computational challenge, the insights gained from optimal strategies in the simplistic setting can be applied to the realistic scenario. For example, in the simplistic setting of ICYF, Pegden et al., 2017 showed that the optimal strategy for the drawing player is to maximize the number of districts they win while maximally "packing" their opponent's voters in the remaining districts. Extending this insight to the realistic setting, the drawing player's strategy can be modeled by a discrete optimization formulation, such as an MIP, that optimizes this dual objective. Even though the resulting strategies may not be optimal, they offer heuristic solutions both players can employ within computational limits. Hence, to simulate the adoption of the protocols in practice, there is a need to model the strategies in a realistic setting and empirically analyze the district maps resulting from either protocol.

The first three parts of this dissertation focus on optimization and game-based approaches for political districting. The first part tackles the issue of modeling fairness, wherein three popular fairness metrics — efficiency gap, partisan asymmetry and competitiveness — are incorporated as objectives within an MIP model designed for PDP. To solve these models using a practical instance of PDP, this part provides a scalable heuristic, called the *multilevel algorithm*, to generate approximate Pareto-optimal district maps, highlighting

the trade-offs between compactness and each of the three political fairness objectives. The second part delves into practical considerations through a case study on congressional districting in Arizona. A local search-based multi-stage heuristic is designed to capture all six redistricting criteria in Arizona's Constitution. Notably, this heuristic can handle the large input size of Arizona's census block adjacency graph, consisting of $155,444$ census blocks. Furthermore, this part conducts a comparative analysis, contrasting district plans derived through the heuristic against Arizona's officially enacted congressional map for the 2021 cycle. The third part focuses on two strategic games: ICYF and bisection protocols. This work extends the strategies from the simplistic to the realistic setting incorporating the discrete and geometric distribution of voters. Each player's strategy is modeled as an MIP, offering heuristic strategies for both players. An empirical analysis is conducted on congressional districting in 18 states to provide empirical insights into the fairness outcomes of the maps generated by the protocols. In summary, these three parts explore the theoretical and computational aspects of integrating fairness within the map-generation process, either in a one-shot optimization setting or in a game setting.

The fourth part of the dissertation transitions from political districting to a broader study of the graph partitioning problem. This part investigates the enforcement of high vertex connectivity in graph partitioning, valued in applications requiring fault tolerance and cohesion among components, such as in power networks and social networks. Specifically, the goal is to ensure that each subgraph resulting from a partition maintains a property called Q-*connectivity* for $Q \geq 2$. A subgraph is considered Q-*connected* if it remains simply connected even when $Q-1$ vertices are removed. Despite the evident advantages of imposing high connectivity requirements in GP, the highly connected graph partitioning (HCGP) problem remains relatively unexplored in the existing literature. Current research in this area is limited to establishing theoretical conditions for the existence of a highly connected partition, particularly for dense input graphs and when the number of parts is not predetermined. Therefore, there is a need for designing mathematical models and solution methods that conceptualize and solve HCGP.

This dissertation establishes a formal mathematical model for HCGP as an Integer Program (IP). This model incorporates Q-connectivity constraints using vertex separators, thereby extending simple connectivity constraints originally developed for PDP. A key computational challenge in solving this model is the exponential number of separator constraints. To tackle this challenge, a branch-and-cut method is provided, which dynamically introduces separator constraints using an efficient integer separation algorithm. Additionally, a heuristic approach is presented to solve large instances, tailored for HCGP with $Q = 2$. A computational analysis of both methods is conducted across instances derived from diverse domains, including power, social and transportation networks. This analysis focuses on the effect of graph sparsity and size on the performance of the two approaches. Moreover, this analysis quantifies the cost of pursuing high connectivity in terms of computational time and the compactness objective. The methods presented in this part contribute to ensuring resiliency and cohesion within graph partitioning.

The rest of the dissertation is organized as follows. Chapter 2 introduces fairness-based optimization models and a multilevel algorithm for political districting. Chapter 3 presents a case study on adopting a local search-based optimization algorithm for congressional districting in Arizona. Chapter 4 provides MIP-based strategies for the bisection and ICYF protocols and a computational analysis using congressional districting instances. Chapter 5 presents a mathematical model for HCGP and efficient solution methods to solve a wide variety of HCGP instances. Collectively, the models and methodologies presented in this dissertation facilitate the incorporation of fairness considerations in political districting and offer tools to incorporate resilience within the broader context of graph partitioning.

Chapter 2

Multi-Objective Optimization for Politically Fair Districting

2.1 Introduction

Fairness in political districting is a topic of subjective debate, with different stakeholders favoring different notions of what constitutes fairness. For example, even if political parties agree that fair districts should have majorities that are "proportional" to their share of voters, the voters themselves may prefer districts that are "competitive" so that their votes have greater impact in representation. Additionally, state districting commissions have increasingly incorporated political fairness criteria into their map-drawing processes. In January 2020, the Governor of Wisconsin signed an Executive Order laying out the criteria for the 2021 districting process, which include a requirement that the proposed plans be "free from partisan bias and partisan advantage" (Office of the Governor, 2019). Several states (such as Arizona, Colorado, New York and Washington) encourage the creation of competitive districts (Mann, 2005). Hence, there is a need for a fairness-based approach to districting that addresses the concerns of multiple stakeholders.

This chapter presents a multi-objective optimization-based districting framework that models several political fairness objectives, thereby extending existing compactness-based models. Three political fairness metrics — *efficiency gap*, *partisan asymmetry*, and *competitiveness* — are considered that explicitly incorporate electoral information. The first two metrics capture fairness among the political parties while competitiveness captures fairness from a voter's perspective. The first metric, the efficiency gap, measures the difference in the "wasted" votes for the two major parties. Minimizing the efficiency gap ensures that one party does not waste significantly more voters than the other, thereby limiting partisan gerrymandering. The second notion of fairness is to ensure that if voter preferences shift in the future, the rate at which the two parties gain or lose seats is *symmetric* between the two parties (assuming the shift is uniform across all the districts). This is accomplished by minimizing the extent of asymmetry, called partisan asymmetry. Third, promoting competitiveness of districts is important to tackle incumbent gerrymandering and to promote greater accountability of representatives to their voters. This is achieved by minimizing the margin of victory in all the districts. The advantages and disadvantages of these metrics have been widely debated in the literature and in judicial proceedings. However, the focus of this chapter is to provide an optimization-based algorithmic framework for a districting process that seeks to measure fairness using these metrics.

The framework presented in this chapter generates (approximate) Pareto-optimal solutions that highlight

the trade-off between compactness and each of the three political fairness measures, namely efficiency gap, partisan asymmetry or competitiveness. Three bi-objective optimization problems are solved using the $\epsilon-$constraint method, each with compactness of districts as the primary objective and a political fairness metric as the secondary objective. To generate problem instance sizes that are solvable by an exact method, the solution framework first solves a series of graph contraction problems forming multiple *levels* of problem instances of increasing *coarseness*. The coarsest level in this multilevel scheme is used to solve the exact bi-objective problem to produce Pareto-optimal solutions. A case study based on congressional districting in Wisconsin with Census 2010 data is presented. The results demonstrate that district plans can be produced that are not only compact but also efficient, symmetric or competitive. The implication of these results is that algorithms that optimize fairness measures are important to design a districting process that integrates the conflicting objectives of multiple stakeholders.

The rest of the chapter is organized as follows. Section 2.2 reviews the key literature in algorithmic districting and in the political debate surrounding fairness in districting. Section 2.3 introduces the political fairness criteria modeled in the multi-objective problem, the optimization formulations, and complexity analysis. Section 2.4 presents a multi-level solution framework that tackles the related computational challenges. Section 2.5 presents computational results based on a case study on congressional districting in Wisconsin, and Section 2.6 provides concluding remarks.

2.2 Related Literature

This chapter links two bodies of literature from (i) political sciences and related fields, and (ii) operations research and management sciences. The former covers extensive work on the socio-political requirements and implications of districting, while the latter focuses on efficient methodologies for creating district plans. The literature in these areas is presented in two parts. Section 2.2.1 discusses how the different notions of political fairness in districting can be quantified, and Section 2.2.2 explores prior research in algorithmic approaches to create district plans.

2.2.1 Quantifying Political Fairness

Political fairness can be approached from the partisan perspective as well as the voters' perspective. Metrics that quantify partisan fairness typically consider *proportionality* and *symmetry*. Proportionality is the notion that a party gets more seats (i.e., wins elections in more districts) if more voters favor that party. An alternate notion of partisan fairness is symmetry, the idea that a district plan favors both the parties symmetrically when vote-shares for the two parties fluctuate in hypothetical future elections. On the other hand, incumbent gerrymandering is a commonly observed (bi)partisan effort at drawing districts that intentionally increases the likelihood that certain (incumbent) candidates win (Silverberg, 1995). From the voters' perspective, the *competitiveness* of a district is a proxy measure against incumbency. Additionally, competition increases accountability of representatives to their constituents (Jones, 2013).

The rest of this section reviews the key literature surrounding the three metrics of fairness considered in this chapter. The advantages and disadvantages of these metrics have been widely debated in the literature and in judicial proceedings. While the focus of this chapter is to provide an optimization-based algorithmic framework for a districting process that seeks to measure fairness using these three metrics, the choice of these metrics is not an endorsement of the metrics themselves, but merely driven by their extensive popularity and

adoption in practical gerrymandering cases. The broader goal of this chapter is to highlight that optimization models can enable a districting commission in achieving a fairness outcome they desire.

Efficiency Gap

The *efficiency gap* (EG) is a popular measure of partisan fairness. Intoduced by Stephanopoulos and McGhee, 2015, EG measures the difference in the "wasted votes" between the two major parties. Here, wasted votes in each district are those votes that are either for the losing party or the surplus votes for the winning party in that district. A fair district plan according to EG is mathematically proportional, i.e., as one party's vote-share increases, the number of districts they win increases proportionally (with a slope of two) (Warrington, 2018). Note that other authors who consider a stricter view on proportionality, where the number of districts won increases at the *same rate* as the vote-share, may consider EG to possess "double-proportionality" (Bernstein and Duchin, 2017), or to be "quasi-proportional" (Chambers et al., 2017). Besides capturing proportionality, Veomett, 2018 shows that EG has an additional term that is a function of the difference in voter turnouts in the different districts. See Appendix A.1 for a detailed discussion on the debate surrounding EG.

Given the extensive academic interest and adaptation of EG as a fairness measure by the courts, this chapter considers the problem of minimizing EG and presents an optimization model to achieve this goal. The model presented in Section 2.3.3 is flexible enough to adapt to alternative ways to compute EG that address at least some of the concerns raised about EG. For example, the issue of uncertainty can be tackled in a robust optimization-based solution method, while the issue of inadvertently creating packed districts can be avoided by setting a hard constraint that enforces a packing limit. The effect of unequal voter turnouts as observed by Veomett, 2018 can be minimized by setting a hard constraint that limits the difference in voter turnout between the districts.

Partisan Asymmetry

Since proportionality is not a constitutional requirement in the U.S., an alternative school of thought seeks the broader notion of partisan symmetry. The symmetry standard requires "the electoral system to treat similarly-situated parties equally, so that each receives the same fraction of legislative seats for a particular vote percentage as the other party would receive if it had received the same percentage [of the vote]" (Grofman, 1983). Several ways to quantify partisan asymmetry have been explored in the literature, with alternatives in the exact numerical function used (Jackman, 1994; Duchin, 2018). A survey by Grofman, 1983 lists eight different ways to measure partisan asymmetry. This chapter considers the measure called *Gini score* which computes a vote-seat curve for each party (i.e., a function of how many districts can be won by a party for every hypothetical vote-share it receives in the state) and then measures the area between the curves for the two parties (DeFord et al., 2021b). See Appendix A.2 for a detailed discussion on the legitimacy of PA as a partisan fairness metric.

Given the historic and extensive presence of symmetry in the debates of partisan fairness, this chapter considers partisan asymmetry as one of the measures of fairness. While LULAC v. Perry, 2006 and DeFord et al., 2021b point to aspects of the symmetry standard that prevent symmetry from being considered a talismanic measure for fairness, the symmetry standard still serves as a strong alternative to viewing fairness through proportionality. Hence, the multi-objective optimization model presented in this chapter enables a districting process that seeks symmetry in tandem with other fairness measures, such as the compactness of districts.

Competitiveness

While both efficiency gap and partisan asymmetry measure fairness with respect to the political parties, competitiveness captures the extent of advantage for incumbent candidates in future elections. In highly competitive districts, small changes in voter preferences lead to larger changes in the seat-share, and are considered desirable in preventing the creation of *safe seats*, or incumbent gerrymandering (Tapp, 2019). Incumbent gerrymandering isolates politicians from being accountable to their constituents, and inhibits communities from receiving "meaningful and fair representation" (Kennedy et al., 2016). Hence, competitiveness captures fairness from the voters' perspective.

Past studies have analyzed the relationship between competitive districts and other metrics. For example, McCarty et al., 2009 note that in the partisan gerrymandering strategy of "cracking" — where one party's voters are divided among many districts where they lose close elections — would technically lead to the construction of competitive districts. Other works show unclear results on the trends in polarized (or non-competitive) districts based on congressional and presidential elections (Abramowitz et al., 2006; McCarty et al., 2009). Regardless, the general consensus is that incumbent gerrymandering can be prevented through the creation of competitive districts, thereby creating greater accountability of candidates to their voters (Friedman and Holden, 2009). This chapter seeks to maximize competitiveness by minimizing the maximum margin of victory in all the districts. The consequence of doing so is that "packing" of voters is minimized.

2.2.2 Algorithmic Approaches to Districting

This section reviews literature in algorithmic approaches to districting. The literature is broadly categorized based on methodology and intent, namely into simulation algorithms, exact optimization methods, and heuristic methods. The goal of a simulation algorithm is to assess whether a given district plan can be considered gerrymandered or not by comparing it with a large number of simulated district plans. On the other hand, an optimization algorithm creates a single district plan that is optimized with respect to a given fairness measure. The intent of a simulation algorithm is to *compare* or *assess*, while the intent of an optimization approach is to *create* or *optimize*. Optimization methods can further be classified into exact and heuristic methods. Exact optimization methods can achieve a guaranteed optimality, but are not scalable for large practical input sizes. Heuristic methods are computationally efficient in finding "good" district plans with respect to a particular fairness measure, but are not be guaranteed to be globally optimal. This section reviews existing work in simulation algorithms, exact optimization methods, heuristic methods, and multilevel algorithms.

Simulation Algorithms

Among the districting algorithms that explicitly capture political fairness metrics, simulation algorithms form the predominant approach. A simulation algorithm generates a large set of district plans using a suitable sampling method with the intent of studying patterns in political objectives, while also serving as a benchmark for existing district plans. For example, Chen and Rodden, 2013 simulate district plans for all U.S. states and conclude that Democratic voters being concentrated in urban areas leads to district plans that strongly favor Republicans. Fifield et al., 2015 provide the first step toward a theoretical framework based on Markov chain Monte-Carlo (MCMC) simulations while incorporating feasibility constraints such as contiguity and population balance. In an MCMC, starting from an initial district plan, a large set of district plans is created by making a series of perturbations. In every step, the conventional method of perturbing a

district plan, called "flip", is to re-assign one unit from its current district to another district (Fifield et al., 2015). The work by DeFord et al., 2021a provide a method called "recombination" for making a more drastic perturbation to a district plan by merging two neighboring districts and completely redrawing their shared boundary. Recombination has a computational advantage since it enables a faster exploration of the solution space of district plans compared to using a flip operation.

Advances in computational power have enabled technologies that can generate a large number of random district plans. Liu et al., 2016 highlight how high-performance parallel computing can be used to generate and analyze more than a million district plans. Even though these algorithms play a vital role in detecting gerrymandering, it is unclear how these methods can be used in the active creation of a district plan that seeks a particular fairness-based outcome. Recent work by Gurnee and Shmoys, 2021, which references an earlier version of the present study, provides a direction toward an optimization-based sampling method which picks the tail-end of the distribution of district plans. However, the optimization modeling of political fairness remains an open problem.

Exact Optimization Methods

Exact optimization methods have traditionally been used to create district plans using non-political objectives such as compactness and population balance (Ricca et al., 2013). Political fairness metrics (e.g., the metrics discussed in Section 2.2.1) have not been explicitly captured in exact models and solution methods. Modeling the political districting problem as a Mixed Integer Program (MIP) with a compactness objective has early roots since the work by Hess et al., 1965. Due to the intractability of the districting problem, finding optimal solutions for large practical instances is a challenge. In particular, contiguity enforcement remains a major challenge. A method used in practice has been to solve the problem without contiguity constraints, and then to use a heuristic to alter a discontiguous solution to satisfy contiguity (Gentry et al., 2015; Salazar-Aguilar et al., 2011). Drexl and Haase, 1999 are the first to provide a set of constraints (albeit exponential in number) that ensure contiguity in districts when using an MIP-based model, while Shirabe, 2009 provides a flow-based model for contiguity enforcement in a unit-allocation problem (a special case of districting). The work by Haase and Müller, 2014 extends the formulation to a sales-force districting problem. More recently, Oehrlein and Haunert, 2017 provide a cutting-plane approach to contiguity constraints and empirically demonstrate its computational advantage over existing formulations, and Validi et al., 2022 provide a stronger branch-and-cut implementation that solves several previously-unsolved instances of compact districting at the census tract level.

In addition to the traditional branch-and-cut based methods, several alternative exact approaches have also been explored. Garfinkel and Nemhauser, 1970 propose a two-stage strategy, where the first stage enumerates all feasible (compact, balanced and contiguous) districts, and the second stage solves a set covering problem to construct an optimal district plan. Mehrotra et al., 1998 present a modified approach: the set covering problem is solved using branch-and-price, and a post-processing stage performs local improvements to the solution.

Despite the extensive research in exact models and methods for districting, political fairness objectives such as efficiency gap, partisan asymmetry and competitiveness have not been captured in MIP models for political districting. It is also unclear if existing computational methods that optimize compactness will be effective when applied to other fairness-based objectives. For example, solving a compactness-seeking model without explicit contiguity constraints typically produces a solution that is either contiguous or "near-contiguous" (Validi et al., 2022). This reinforcing relationship between contiguity and compactness is advantageous when

using a cutting planes approach by Oehrlein and Haunert, 2017 or Validi et al., 2022 since "few inequalities are needed to prove optimality" (Validi et al., 2022). However, when solving a MIP with political fairness-based objectives/constraints, it is unclear whether these approaches will be as computationally effective.

The issue of fairness in districting has been studied in designing game theoretical protocols. In such a game, two or more political parties/candidates strategize to maximize their desired outcomes such as the numbers of districts they win (Landau and Su, 2014; Pegden et al., 2017; De Silva et al., 2018; Benadè et al., 2021). The goal of these works is to design a game such that the district plan produced is "fair" according to some measure(s). This goal is fundamentally different from the goal of this chapter, which is to optimize a specific fairness objective(s) where that a single decision maker (such as an independent redistricting commission) draws the district plan.

Heuristic Methods

Heuristic algorithms are popular in large-scale practical districting applications (Ricca and Simeone, 2008). The heuristics explored in the literature predominantly cater to the goal of optimizing non-political objectives such as compactness and population balance. Some of these methods include simulated annealing (Browdy, 1990; D'Amico et al., 2002), tabu search (Bozkaya et al., 2003), local search (Ricca and Simeone, 2008), genetic algorithm (Bacao et al., 2005), and polygonal clustering (Joshi et al., 2012), among others. Computational geometry-based techniques using Voronoi regions have also been explored (Ricca et al., 2008).

Some heuristic approaches have explored political objectives. For example, Chatterjee et al., 2020 and King et al., 2015 propose local search methods that optimize the efficiency gap and competitiveness metrics, respectively. These methods improve a given district plan using a sequence of neighborhood perturbations. While these political objectives have been explored in heuristic optimization, they have not been explored in exact optimization algorithms. Additionally, the partisan asymmetry metric has not been incorporated in an optimization setting. Hence, there is an opportunity for research in the integration of optimization methods in political fairness-based districting.

Multilevel Algorithms

A multilevel algorithm is a framework for integrating an exact optimization method within a practical heuristic capable of solving large instances. They are considered the most effective algorithms for graph partitioning problems with applications in VLSI design and image processing (Buluç et al., 2016). A multilevel algorithm creates multiple hierarchical levels of graphs by an iterative coarsening procedure (where adjacent nodes are merged to create a smaller graph) and then plans an optimal district solution from the coarsest graph instance back to the original graph (Kernighan and Lin, 1970; Hendrickson and Leland, 1995). Note that the optimal solution found using the coarsest instance when mapped back to the original instance may not be optimal to the original instance, and hence this algorithm is a heuristic method to solve an optimization problem. This framework has also been adopted to other combinatorial optimization problems such as the traveling salesman problem (Walshaw, 2002) and graph-drawing (Walshaw, 2000). While districting problems share many features with graph partitioning problems, they also differ from graph partitioning problems (whose objective typically is to minimize the total weight of the edges cut by the partition) in that districting involves more constraints, including contiguity and tighter population balance constraints, in addition to a broader set of fairness objectives. Recent work by Magleby and Mosesson, 2018 presents a multilevel algorithm to generate random district plans where the coarsening stage randomly chooses adjacent units

to be merged. This chapter investigates how a multilevel algorithm can be adapted to solve deterministic large-scale districting problems, especially when considering multiple objectives.

2.3 Multi-objective Politically Fair Districting

This chapter considers districting as an optimization problem with four objectives: compactness, efficiency gap, partisan asymmetry and the competitiveness of districts. The general multi-objective districting model is introduced in Section 2.3.1. The background, formulation and analysis of each of the single-objective optimization problems are presented in the rest of this section.

2.3.1 The Districting Problem

A mathematical model for districting for political representation is now formalized. A *Districting Problem* (DP) partitions a geographical region into a finite number of districts where each district satisfies certain requirements. The input to DP consists of a set of discrete geographical units V (or simply *units*), their adjacency information, and the number of districts K. Each unit represents an area of administrative significance such as a county, census tract, or census block. Two units $i, j \in V$ are said to be *adjacent* if they share a boundary curve which contains a continuous segment (as opposed to being a set of singleton points). An *adjacency graph* $G = (V, E)$ is an undirected graph where the set of nodes V represents the set of units, and an edge (i, j) exists in the edge set E if and only if units i and j are adjacent. Note that an edge (i, j) here denotes an un-ordered pair of nodes, i.e., (i, j) is equivalent to (j, i). For each unit $i \in V$, let $N(i) \coloneqq \{j \in V : (i, j) \in E\}$ denote the set of *neighbors* of i. Additionally, each unit $i \in V$ contains a certain population of residents, denoted by $p_i \geq 0$. A *district* is a subset of V that satisfies certain conditions, and a *district plan* is a partition of V into K districts, i.e., $z : [K] \to 2^{|V|}$ denotes a district plan where for each $k \in [K]$, $z(k) \subset V$ is a district. A district plan z is said to be feasible for DP if for each district $k \in [K]$, the subgraph induced by $z(k)$ in G is connected, and the population $\sum_{i \in z(k)} p_i$ belongs to $[lb, ub]$, where lb and ub are the lower and upper bounds on the population required in each district, respectively. Typically, these bounds are formulated as $[\overline{P}(1-\tau), \overline{P}(1+\tau)]$, where $\overline{P} \coloneqq (\sum_{i \in V} p_i)/K$ is an ideal population in each district, and $\tau \geq 0$ is the population deviation tolerance. Given an objective function $\phi : [K] \times 2^{|V|} \to \mathbb{R}$, DP finds a feasible district plan that minimizes ϕ.

The *Politically Fair Districting Problem* (PFDP) is a multi-objective generalization of DP, with each objective representing a particular notion of fairness. In addition to the inputs for DP, for district plans to be evaluated on their political fairness, PFDP also requires information on the political leanings of voters. Consider a two-party electoral system, where parties A and B have a distribution of voters spread among the units in V. Let $p_i^A, p_i^B \geq 0$ (with $p_i^A + p_i^B \leq p_i$) be the number of voters for parties A and B respectively residing in unit $i \in V$. For a given district plan z, in each district $k \in [K]$, let $P_k^r(z) \coloneqq \sum_{i \in z(k)} p_i^r$ be the number of voters for party $r \in \{A, B\}$. In a plurality-based election (e.g., in U.S. congressional elections), a party A (B) is said to *win* (*lose*) district k if $P_k^A(z) > P_k^B(z)$, i.e., if A has more voters in district k than party B. Note that ties (i.e., $P_k^A(z) = P_k^B(z)$ for some $k \in [K]$) rarely occur in practice when the number of voters is large and the units are discrete with a heterogeneous distribution of voters. For a given district plan, based on the political leanings of the voters in each district and the districts won/lost by the two parties, one can assess how politically *fair* the district plan is. There are several ways to assess political fairness (e.g., the efficiency gap and partisan asymmetry) as discussed in Section 2.2.1. Let $\{\phi_q\}_{q=1}^{\mathcal{O}}$ be a set of $\mathcal{O} \geq 1$ political fairness objective functions. Here, for each $q \in [\mathcal{O}]$, $\phi_q : [K] \times 2^{|V|} \to \mathbb{R}$ represents a real-valued function that

measures the fairness of a district plan z. PFDP is an $\mathcal{O}-$objective optimization problem that minimizes the set of objective functions $\{\phi_q\}_{q=1}^{\mathcal{O}}$ within the solution space of DP.

This section presents a mathematical model for a PFDP with $\mathcal{O} = 4$ objectives, namely compactness, efficiency gap, partisan asymmetry, and competitiveness. Computational experiments applying this framework to instances with select pairs of these objectives (i.e., $\mathcal{O} = 2$) are presented in Section 2.5.

PFDP can be formulated as follows. The primary decision variables that assign units to districts are given by,

$$x_{ij} = \begin{cases} 1, & \text{if } j\ (\neq i) \in V \text{ is assigned to the district with center } i \in V. \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ii} = \begin{cases} 1, & \text{if unit } i \in V \text{ is chosen as a district center.} \\ 0, & \text{otherwise.} \end{cases}$$

Here, a *district center* is a special unit in a district that serves as the source of flow within its district in the flow-based contiguity constraints discussed below. Originally introduced by Shirabe, 2009, flow-based constraints for contiguity have recently become popular for contiguity enforcement in MIPs. Originating from every unit $i \in V$, let $f_{ijv} \geq 0$ be a decision variable that denotes the amount of flow from unit $j \in V$ to its neighbor $v \in N(j)$, whose value is positive only if i is a district center, and units v and j are both assigned to the district with center i. Adapting from the model presented in seminal work by Hess et al., 1965, PFDP can be formulated as,

$$\text{(PFDP) Minimize } \{\phi_1(x), \phi_2(x), \ldots, \phi_{\mathcal{O}}(x)\} \tag{2.1}$$

$$\text{subject to } (1-\tau)\overline{P}\ x_{ii} \leq \sum_{j \in V} x_{ij} p_j \leq (1+\tau)\overline{P}\ x_{ii} \quad \forall\ i \in V, \tag{2.2}$$

$$\sum_{i \in V} x_{ii} = K, \tag{2.3}$$

$$\sum_{i \in V} x_{ij} = 1 \quad \forall\ j \in V, \tag{2.4}$$

$$x_{ij} \leq x_{ii} \quad \forall\ i, j \in V, \tag{2.5}$$

$$x_{ij} + \sum_{v \in N(j)} (f_{ijv} - f_{ivj}) = 0 \quad \forall\ i, (\neq) j \in V, \tag{2.6}$$

$$x_{ii} + \sum_{v \in N(i)} (f_{iiv} - f_{ivi}) - \sum_{v \in V} x_{iv} = 0 \quad \forall\ i \in V, \tag{2.7}$$

$$|V|\ x_{ij} - \sum_{v \in N(j)} f_{ivj} \geq 0 \quad \forall\ i, j \in V, \tag{2.8}$$

$$x_{ij} \in \{0, 1\},\ f_{ivj} \geq 0 \quad \forall\ i, j \in V, v \in N(j). \tag{2.9}$$

The tuple of objectives in (2.1) are minimized in a PFDP. Constraints (2.2) define the upper and lower bounds on district populations. Constraint (2.3) ensures that there are exactly K districts. Constraints (2.4) assign a unique district to each unit. Constraints (2.5) ensure that a unit can be assigned to a district with unit i as its center only if i is district center, i.e, $x_{ii} = 1$. Constraints (2.6)-(2.8) have been adapted from the flow-conservation constraints introduced in Haase and Müller, 2014 for a sales force deployment problem. Here, flow originates from a district center i, and every unit j assigned to the district with center i acts as a sink for one unit of flow. These constraints ensure that overall flow is conserved if and only

if all the districts are contiguous. Note that Oehrlein and Haunert, 2017 provide a more elegant way to express the flow constraints, where constraints (2.7) are omitted. However, the computational study in this chapter includes constraints (2.7) since preliminary investigations revealed that the MIP can be solved faster with their inclusion than without them. Finally, constraints (2.9) define the variable domains. This formulation can be improved with the inclusion of the constraints $\sum_{v \in N(i)} f_{ivi} = 0$ for all $i \in V$. Additionally, an alternative formulation with an x_{ik} variable that assigns unit $i \in V$ to district $k \in [K]$ could reduce the number of variables, though preliminary analysis indicates that this formulation has weaker relaxation bounds (compared to using the x_{ij} variable), which lead to a slower convergence.

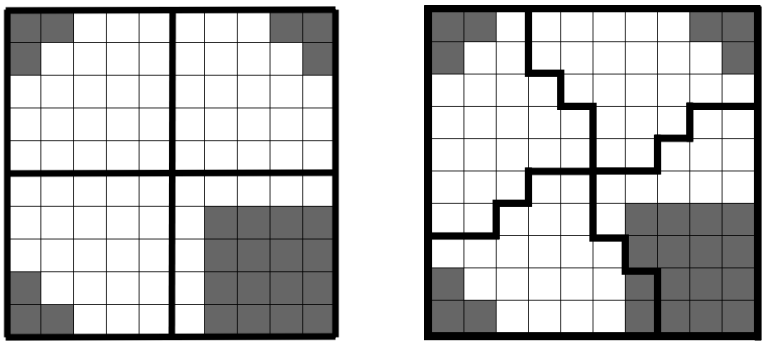
2.3.2 Compact Districting Problem

Compactness evaluates a district's geometrical shape and can be quantified in several ways (Young, 1988). A popular metric is the *moment of inertia*, measured by the squared sum of distances from the units to their geographical district centers, weighted by the unit populations. Here, a district center is a unit with the smallest sum of weighted squared distances to the rest of the units in its district, and whose location is determined within the optimization algorithm. Note that this district center still serves as the source of flow in constraints (2.6)-(2.8). Moment of inertia has been used in compact districting since early work by Hess et al., 1965. Let $d_{ij} \geq 0$ be the given distances between every pair of units $i, j \in V$. The *Compact Districting Problem*, or Compact-DP, finds a district plan that minimizes this compactness objective, and can be formulated as,

$$\text{(Compact-DP) Minimize } \phi_{comp}(x) \coloneqq \sum_{i,j \in V} p_j d_{ij}^2 x_{ij} \tag{2.10}$$
$$\text{subject to (2.2)} - \text{(2.9)}.$$

The compactness objective ϕ_{comp} in (2.10) minimizes the moment-of-inertia function. Compact-DP is a well studied classical problem that generalizes the NP-complete p-median problem (Megiddo and Supowit, 1984).

Even though compact districts are generally perceived to be *fair*, optimally compact districts could be considered unfair upon consideration of voter information. Consider an example with units forming a 10×10 grid (Figure 2.1), where each 1×1 square is a unit. Assume that the shaded units are densely populated with party A voters, whereas the unshaded units are populated with party B voters and non-voters, i.e., the tuple (p_i, p_i^A, p_i^B) for each unit i is: $(40, 40, 0)$ for the shaded units and $(40, 0, 10)$ for the unshaded units. Here, the total number of voters for A and B are 1000 and 750, respectively. This example illustrates a scenario of spatial polarization of voters in a real-life setting where different areas could have different concentrations of party support (Chen and Rodden, 2013). Consider the four districts defined by the four equal-sized quadrants of the grid in Figure 2.1a. Clearly, these districts satisfy the contiguity and population balance constraints, and are also optimally compact. However, the seat-share for B is 0.75 (i.e., B wins three out of the four districts), while their vote-share is 0.43 (i.e., B recieves 750 out of the $1,750$ voters). From a proportionality stand-point, this district plan is not considered fair since B wins three times the number of districts as A does, even though B's vote-share is less than A's. In contrast, in the district plan in Figure 2.1b, the seat-share for B is 0.5, which is closer to B's vote-share than that of the previous district plan despite having less compact districts. Hence, if proportionality is sought in districting, there is a need to explicitly consider voter information in an optimization model in addition to the compactness objective.



(a) Optimal district plan to Compact-DP

(b) Districts with proportional seat-share

Figure 2.1: Example with four districts on a grid graph suggesting that an optimally compact solution may not yield a proportional seats-share to the two parties.

2.3.3 Efficient Districting Problem

In a scenario where one party (A) gerrymanders (i.e., intentionally draws the districts to gain artificial advantage in the district elections), the phenomenon of *packing* and *cracking* is observed. Packing occurs when a large number of the other party's (B) voters are assigned to a small number of districts in which party B wins by large margins, while cracking occurs when the remaining party B voters are *diluted* among the rest of the districts in which party A holds a majority. Even though this phenomenon has been widely discussed in decades of gerrymandering research, it was not until recently that a metric to quantify packing and cracking has been formulated and adopted in practice. The *efficiency gap* introduced by Stephanopoulos and McGhee, 2015 captures both packing and cracking by counting the difference in wasted votes between the two parties. A vote in a certain district is said to be *wasted* if it is either a surplus vote for the winning party, or a vote to the losing party. In each district $k \in [K]$, let $w_k^A(z)$ be the wasted votes for party A given by,

$$w_k^A(z) = \begin{cases} P_k^A(z) - \dfrac{P_k^A(z) + P_k^B(z)}{2}, & \text{if } P_k^A(z) > P_k^B(z). \\ P_k^A(z), & \text{if } P_k^A(z) < P_k^B(z). \end{cases}$$

Exactly half the votes in every election are wasted. However, if one party disproportionately wastes more votes than the other, it is considered to be an indication of partisan gerrymandering in practical districting scenarios (Stephanopoulos and McGhee, 2015), unless one party achieves an overwhelming majority of the total votes (Chambers et al., 2017). Among the wasted votes, the surplus votes for the winning party quantify packing, while all the votes for the losing party quantify cracking. Note that when there is a tie in a district k (i.e., when $P_k^A(z) = P_k^B(z)$), the notion of wasted votes is not defined in Stephanopoulos and McGhee, 2015 since there is no clear winner in that district. Even though ties rarely occur in practice as noted in Section 2.3.1, the model discussed later in this section explicitly deals with ties. Assuming there are no ties, the difference in wasted votes between the two parties in each district k is given by,

$$w_k^A(z) - w_k^B(z) = \begin{cases} \dfrac{P_k^A(z) - 3P_k^B(z)}{2}, & \text{if } P_k^A(z) > P_k^B(z). \\ \dfrac{3P_k^A(z) - P_k^B(z)}{2}, & \text{if } P_k^A(z) < P_k^B(z). \end{cases} \tag{2.11}$$

Party A wastes more votes than party B when $w_k^A(z) - w_k^B(z)$ is positive; vice versa if it is negative. For a given district plan z, the *efficiency gap* (ϕ_{EG}) measures the magnitude of the net difference in wasted votes between two political parties A and B across all the districts normalized by the total number of voters, defined as,

$$\phi_{EG}(z) \coloneqq \frac{\left|\sum_{k=1}^{K}(w_k^A(z) - w_k^B(z))\right|}{\sum_{k=1}^{K}(P_k^A(z) + P_k^B(z))}. \tag{2.12}$$

For any given district plan z, $\phi_{EG}(z)$ is a value between 0 and 0.5, where a value of 0 implies that both parties have wasted an equal number of votes. In this chapter, a district plan is said to be *perfectly efficient* if both the parties waste the same number of votes, i.e., with efficiency gap equal to zero. The *Efficient Districting Problem*, or Efficient-DP, is a districting problem whose objective is to minimize the efficiency gap, $\phi_{EG}(z)$. Recent work by Chatterjee et al., 2020 shows that minimizing the efficiency gap with population balance constraints is strongly NP-complete when the instance given by the adjacency graph is planar (which is typically the case for practical districting problems).

This section presents an optimization model for Efficient-DP. In this model, a key assumption is that in the rare case of a tie in a district k (i.e., when $P_k^A(z) = P_k^B(z)$), the model arbitrarily chooses a winner in that district such that this choice minimizes the efficiency gap. Efficient-DP can be formulated as a mixed integer program (MIP). Let $a_i \in \mathbb{R}$ for unit $i \in V$ be the difference in wasted votes between parties A and B in a district with center i; 0 if i is not a district center. Additional decision variables supporting this formulation are given by,

$$y_i^A = \begin{cases} 1, & \text{if } i \in V \text{ is a district center and party } A \text{ wins that district.} \\ 0, & \text{otherwise.} \end{cases}$$

$$v_{ij}^A = \begin{cases} 1, & \text{if party } A \text{ wins the district with center } i, \text{ and } j \in V \text{ is assigned to that district.} \\ 0, & \text{otherwise.} \end{cases}$$

Efficient-DP can be formulated as,

$$\text{(Efficient-DP) Minimize } \phi_{EG}(x) = \frac{|\sum_{i\in V} a_i|}{\sum_{i\in V}(p_i^A + p_i^B)} \tag{2.13}$$

$$x \in \text{HESS}(2.2) - (2.9),$$

$$-M \le \sum_{j\in V}(p_j^A - p_j^B)x_{ij} - My_i^A \le 0 \quad \forall\, i \in V, \tag{2.14}$$

$$v_{ij}^A \le x_{ij} \quad \forall\, i,j \in V, \tag{2.15}$$

$$v_{ij}^A \le y_i^A \quad \forall\, i,j \in V, \tag{2.16}$$

$$v_{ij}^A \ge x_{ij} + y_i^A - x_{ii} \quad \forall\, i,j \in V, \tag{2.17}$$

$$y_i^A \le x_{ii} \quad \forall\, i \in V, \tag{2.18}$$

$$a_i = \sum_{j\in V}\Big(\frac{3p_j^A - p_j^B}{2}\Big)x_{ij} - \sum_{j\in V}(p_j^A + p_j^B)v_{ij}^A \quad \forall\, i \in V, \tag{2.19}$$

$$v_{ij}^A, y_i^A \in \{0,1\},\ a_i \in \mathbb{R} \quad \forall\, i,j \in V. \tag{2.20}$$

The efficiency gap objective ϕ_{EG} in (2.13) minimizes the absolute value of the net difference in wasted votes normalized by the total number of voters. The absolute value function in (2.13) is linearized in constraints

(2.35) in Section 2.4.2. Note that the denominator in (2.13) is a constant which can be removed in order to make the model more numerically stable for an MIP solver. Constraints (2.14) define $\{y_i^A\}_{i\in V}$ in relation to $\{x_{ij}\}_{i,j\in V}$, where M can be any value greater than $\overline{P}(1+\tau)$. Here, the first inequality ensures that $y_i^A = 1 \Rightarrow \sum_{j\in V}(p_j^A - p_j^B)x_{ij} \geq 0$. The second inequality ensures that $\sum_{j\in V}(p_j^A - p_j^B)x_{ij} \geq 0 \Rightarrow y_i^A = 1$. Constraints (2.15)-(2.17) linearize the quadratic constraints $v_{ij}^A = x_{ij}y_i^A$ that ensure that $v_{ij}^A = 1$ if and only if $x_{ij} = y_i^A = 1$ using standard linearization techniques (Adams and Sherali, 1990). Constraints (2.18) ensure that $y_i^A = 1$ only if unit i is a district center. Constraints (2.19) define the difference in wasted votes $\{a_i\}_{i\in V}$. If $i \in V$ is a district center (i.e., $x_{ii} = 1$), let $V^i \subset V$ be the set of units assigned to that district (i.e., $V^i \coloneqq \{j \in V : x_{ij} = 1\}$). For each $i \in V$, constraint (2.19) ensures that $a_i = \sum_{j\in V^i}(p_j^A - 3p_j^B)/2$ if A wins the district with center i, or $a_i = \sum_{j\in V^i}(3p_j^A - p_j^B)/2$ if B wins that district. Hence, in accordance with (2.11), a_i stores the difference in wasted votes between the two parties in the district with center i. If i is not a district center, then a_i is 0. Constraints (2.20) define the variable domains.

The way in which the model picks the winner in each district with center $i \in V$ is detailed below. First consider the case when $\sum_{j\in V}(p_j^A - p_j^B)x_{ij} \neq 0$. Then, i is a district center and there is no tie in that district. In this case, there are two possibilities: (i) $\sum_{j\in V}(p_j^A - p_j^B)x_{ij} > 0$ (where party A wins that district) or (ii) $\sum_{j\in V}(p_j^A - p_j^B)x_{ij} < 0$ (where party B wins that district). Then, constraint (2.14) ensures that $y_i^A = 1$ in case (i) and $y_i^A = 0$ in case (ii), satisfying the definition of y_i^A. Second, consider the case when $\sum_{j\in V}(p_j^A - p_j^B)x_{ij} = 0$. Then, either i is not a district center or there is a tie in the district with center i. If i is not a district center, $x_{ii} = 0$ and hence constraint (2.18) ensures that $y_i^A = 0$. In the case when there is a tie, according to constraint (14), y_i^A can either be 0 or 1. The rest of the constraints (2.15)-(2.19) define the wasted votes a_i in district i. Hence, the optimization model selects the value of y_i^A appropriately such that the wasted votes are attributed to that party that minimizes the EG objective. In particular, when the model sets y_i^A to be 1(0), party A (B) wins that district and all of B (A)'s voters are set to be wasted. The end result is that the solver will break ties in a way that minimizes the overall efficiency gap.

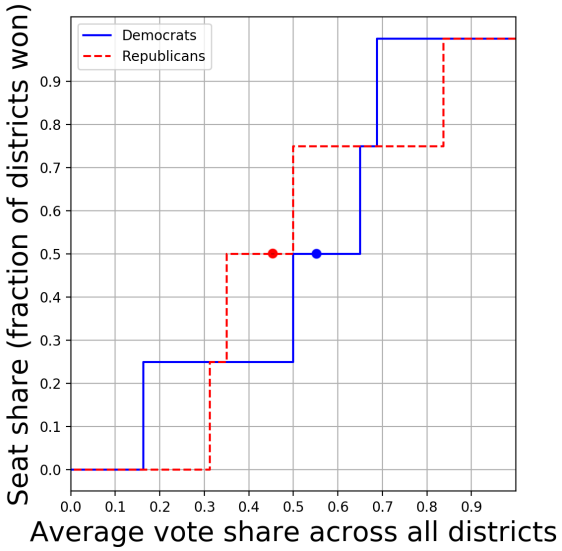
2.3.4 Symmetric Districting Problem

A well-established notion of partisan fairness in a two-party system is *partisan asymmetry*, which captures the extent of advantage one party has over the other when voter preferences fluctuate. To understand the idea behind partisan asymmetry, a *vote-seat curve* is first defined. For a given district plan, let $\omega_k \in [0,1]$ be the district vote-share in district $k \in [K]$ for an arbitrary party A. Let $\overline{\omega} \coloneqq (\sum_{k\in[K]}\omega_k)/K$ be the *average vote-share* for A across all the districts. For every given average vote-share $\overline{\omega} \in [0,1]$, the *vote-seat curve* is defined to be the corresponding fraction of districts won by A, denoted by $s(\overline{\omega}) \in [0,1]$. To compute the vote-seat curve for a given district plan, starting from A's actual vote-share (based on past elections), its vote-shares across all the districts are shifted by adding and subtracting votes. A key assumption here is that the shifting is done *uniformly* across all the districts, as discussed in Duchin, 2018 and Katz et al., 2020. Then, the vote-seat curve for party A is a step function that gains a step when A wins a district. Party B's vote-seat curve is given by $1 - s(1-\overline{\omega})$. Note that prior work in defining vote-seat curves consider each party's *overall vote-share*, which is the fraction of votes for a party across the entire state, instead of the average vote-share. However, it was empirically observed that the average vote-share is close to the overall vote-share in the district plans obtained from real-world instances. For example, in the 2010 congressional district plan for Wisconsin, Democrats (D) have an average vote-share of 0.513, whereas their overall vote-share in the state is 0.516. Across the U.S. states with at least two congressional districts in each, D's average vote-share in each state's 2010 congressional district plan and D's overall vote-share in that state differ by 0.004 on

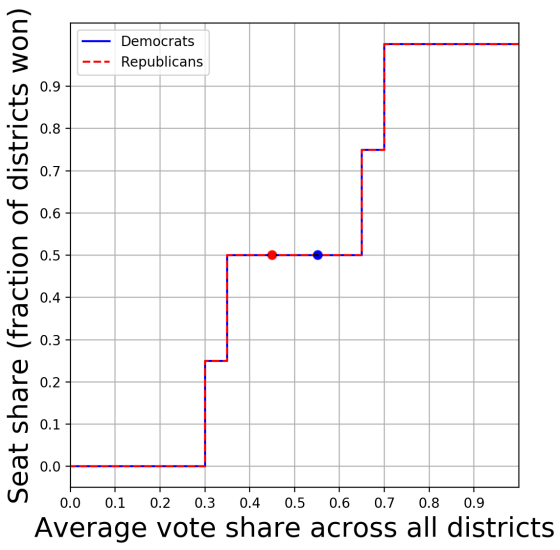
average. Additionally, considering the average vote-share has computational advantages since it is a linear function of the district vote-shares.

Partisan asymmetry quantifies the functional difference between the vote-seat curves for the two parties, given by $|s(\overline{\omega}) - (1 - s(1 - \overline{\omega}))|$. As proposed by Grofman, 1983 as "Measure 7", this chapter measures partisan asymmetry for a district plan z by the area between the two curves. This measure is also referred to as the *partisan Gini score* in DeFord et al., 2021b given by,

$$\phi_{PA}(z) \coloneqq \frac{1}{K} \int_0^1 |s(\overline{\omega}) - (1 - s(1 - \overline{\omega}))| d\overline{\omega}. \tag{2.21}$$



(a) For plan $z^{(1)}$ with initial district vote-shares $\{0.35, 0.4, 0.55, 0.9\}$.



(b) For plan $z^{(2)}$ with initial district vote-shares $\{0.35, 0.4, 0.7, 0.75\}$

Figure 2.2: The vote-seat curves for an example with four districts and Democrats' average vote-share of 0.55.

To illustrate the computation of partisan asymmetry, consider an example with four districts and two parties — the Democrats (D) and the Republicans (R). For a given district plan $z^{(1)}$, let D's actual vote-shares in the four districts be $\{0.35, 0.4, 0.55, 0.9\}$, where D wins two districts. Here, D's actual average vote-share is $\overline{\omega} = 0.55$. To start computing D's vote-seat curve, their district vote-shares are incremented uniformly across all the districts until D *just* wins one more district. Doing so brings D's district vote-shares to $\{0.45, 0.5, 0.65, 1\}$ and the corresponding average vote-share to $\overline{\omega} = 0.65$. Similarly, as votes are uniformly first incremented (and then decremented) across the districts until D wins (loses) all the districts, the points at which D just wins (and loses) seats are recorded to obtain its vote-seat curve $\{s(\overline{\omega}) : \overline{\omega} \in [0, 1]\}$. Note that when a district vote-share reaches 1 (0), votes in that district are no longer incremented (decremented). Figure 2.2a depicts the vote-seat curves for D and R. The area between the two curves gives plan $z^{(1)}$ a partisan asymmetry of $\phi_{PA}(z^{(1)}) = 0.04$. Consider another district plan $z^{(2)}$, where D's actual district vote-shares in the four districts are $\{0.35, 0.4, 0.7, 0.75\}$, where D wins two districts. The vote-seat curves for D and R are depicted in Figure 2.2b and this plan has a partisan asymmetry of $\phi_{PA}(z^{(2)}) = 0$.

The asymmetry in plan $z^{(1)}$ can be interpreted as follows. If R's average vote-share increases (from its actual value of 0.45) to 0.55, R wins three districts; whereas D wins only two for the same average

vote-share. In general, in the regions in the vote-seat space where one party's seat-share is higher than the other party's, the difference in the seats is considered to be asymmetry in outcomes between the two parties. This asymmetry is intrinsic to how the voters are distributed in the districts, and is measured by the area between the curves. On the other hand, the vote-seat curves for plan $z^{(2)}$ indicate that both parties win the same number of districts for the same hypothetical vote-shares and is hence perfectly symmetric.

The *Symmetric Districting Problem*, or Symmetric-DP, is a districting problem with two parties A and B whose objective minimizes the partisan asymmetry ϕ_{PA} of a district plan. The following variables are introduced to formalize Symmetric-DP as a Mixed Integer Non-linear Program.

$\omega_i =$ district vote-share for party A in the district with center $i \in V$; 0 if i is not a district center.
$\alpha_k =$ the k^{th} largest district vote-share for party A, for $k \in [K]$
$\mu_{km} =$ shifted district vote-share in district $m \in [K]$ when party A wins $k \in [K]$ districts
$\hat{\omega}_k =$ minimum average vote-share for A to win $k \in [K]$ districts.

Note that since the two vote-seat curves are step functions, the objective ϕ_{PA} can be expressed as a finite sum over the regions defined by the K breakpoints of each curve. Here, a breakpoint in the step function corresponds to the average vote-share when a party *just* wins one more district. Further, for all $k \in [K]$, let $k\max\{\omega_i\}_{i\in V}$ denote a function that returns the k^{th} largest value from the set of values $\{\omega_i\}_{i\in V}$. Then, Symmetric-DP can be formalized as,

$$\text{(Symmetric-DP) Minimize } \phi_{PA}(x) = \frac{1}{K}\sum_{k=1}^{K} |\hat{\omega}_k - (1 - \hat{\omega}_{K-k+1})| \tag{2.22}$$

$$\text{subject to } (2.2) - (2.9),$$

$$\omega_i = \begin{cases} \dfrac{\sum_{j\in V} p_j^A x_{ij}}{\sum_{j\in V}(p_j^A + p_j^B)x_{ij}}, & \text{if } x_{ii} = 1 \\ 0, & \text{otherwise} \end{cases} \quad \forall\ i \in V, \tag{2.23}$$

$$\alpha_k = k\max\{\omega_i\}_{i\in V} \quad \forall\ k \in [K], \tag{2.24}$$

$$\mu_{km} = \begin{cases} 0, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \le 0 \\ \alpha_m + \frac{1}{2} - \alpha_k, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \in (0,1) \\ 1, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \ge 1 \end{cases} \quad \forall\ k, m \in [K], \tag{2.25}$$

$$\hat{\omega}_k = \frac{1}{K}\sum_{m=1}^{K} \mu_{km} \quad \forall\ k \in [K], \tag{2.26}$$

$$\mu_{km}, \omega_i, \alpha_k, \hat{\omega}_k \in [0,1] \quad \forall\ i \in V,\ k, m \in [K]. \tag{2.27}$$

The objective in (2.22) minimizes the partisan asymmetry function. Each constraint in (2.23) defines ω_i to be A's district vote-share if $i \in V$ is a district center; 0 otherwise. Constraints (2.24) define the k^{th} largest vote-share α_k for each $k \in [K]$, where the $k\max$ function returns the $k-$th largest value of a given set. Each constraint in (2.25) defines the shifted district vote-share μ_{km} in district m when party A wins k districts. Here, the district vote-share α_m in each district m is altered by a value of $\alpha_k - \frac{1}{2}$, which is the fraction of voters needed to *just* win k districts. This piecewise linear relationship ensures that the shifted district vote-share $\left(\alpha_m - \left(\alpha_k - \frac{1}{2}\right)\right)$ does not exceed the $[0,1]$ interval. Each constraint in (2.26) defines the minimum average vote-share $\hat{\omega}_k$ needed for A to win $k \in [K]$ districts. Constraints (2.27) restrict the variables to be

non-negative. The non-linear constraints (2.23)-(2.25) can be linearized, as described in Proposition 1.

Proposition 1. *Symmetric-DP can be formulated as a Mixed Integer Linear Program (MILP).*

The proof given in the appendix A.4 linearizes the constraints (2.23)-(2.25) by the introduction of additional variables and constraints. The proof also presents more details on the intuition behind constraints (2.23)-(2.25). Note that the objective function in (2.22) can be linearized by standard linearization techniques and is implemented in constraint form as discussed in Section 2.4.2.

2.3.5 Competitive Districting Problem

Incumbent gerrymandering (or bipartisan gerrymandering) is the act of intentionally protecting incumbent candidates by drawing districts that can be considered "safe" for them, i.e., highly packed in the incumbents' favor (Mann, 2005). Avoiding the creation of safe districts, or in turn creating *competitive* districts, is vital to ensure that the representatives are responsive to their constituents. In the U.S., Arizona and Colorado have explicitly encoded in their Constitutions that districts be competitive "to the extent practicable" without significant detriment to other districting goals, while Washington and New York require that competition is "encouraged" or at the least "not discouraged"; eleven other states prohibit the creation of districts that intentionally favor incumbent candidates (National Conference of State Legislatures, 2020).

In a two-party system, the competitiveness of a district is quantified by the *margin of victory*, the fraction of voters by which the winning party leads the other party (McDonald, 2006). The smaller the margin, the more competitive the district is. Given a district plan, the competitiveness of the plan can also be measured by the number of "competitive districts", where a district is competitive if its margin of victory is below a certain threshold (McDonald, 2006). However, when an optimization algorithm maximizes the number of competitive districts, it frequently creates district plans that "pack" the non-competitive districts. For example, an optimal solution with six competitive districts out of eight often has the other two districts be highly packed. Hence, to ensure that all the districts are as competitive as possible, this chapter measures competitiveness by minimizing the maximum margin across all the districts. For a district plan z, the competitiveness objective, $\phi_{cmpttv}(z)$, is defined to be the maximum margin among all the districts, i.e.,

$$\phi_{cmpttv}(z) \coloneqq \max_{k\in[K]} \frac{|P_k^A(z) - P_k^B(z)|}{P_k^A(z) + P_k^B(z)}.$$

The *Competitive Districting Problem*, or Competitive-DP, is a districting problem whose objective is to minimize the competitiveness function $\phi_{cmpttv}(z)$. Minimizing the maximum margin encourages the creation of competitive districts by associating the quality of a district plan by its least-competitive district.

Even though the computational hardness of Competitive-DP is unclear, this chapter shows hardness results for a closely related problem. The *Aggregate Competitive Districting Problem* (ACDP) finds a feasible district plan z that minimizes the maximum absolute difference in the two parties' voters given by $\max_{k\in[K]} |P_k^A(z) - P_k^B(z)|$. ACDP is different from Competitive-DP in that ACDP's objective does not contain the normalization factor in Competitive-DP's objective given by the total number of voters in that district. Despite this difference, the ACDP's objective captures the essence of creating competitive districts. It can be shown that ACDP is NP-complete if $K = 2$ and is strongly NP-complete if $K \geq 3$ with or without the population balance constraints (2.2).

Theorem 1. *For a general graph $G = (V, E)$, unit populations $\{p_i, p_i^A, p_i^B\}_{i \in V}$ and number of districts $K \geq 2$, the Aggregate Competitive Districting Problem (ACDP) with or without the population balance constraints is NP-complete if $K = 2$ and is strongly NP-complete if $K \geq 3$.*

The proof (presented in Section A.3 of the appendix) proceeds by showing a reduction from the NP-complete 2-partitioning problem for $K = 2$, and from the strongly NP-complete 3-partitioning problem for $K \geq 3$ (Garey and Johnson, 1978).

Competitive-DP can be formulated as the following MIP by linearizing the fractional and absolute value functions in the objective. Let $h \geq 0$ be the maximum margin across all the districts. For every pair of units $i, j \in V$, let $b_{ij} \coloneqq h\ x_{ij}$ be an artificial quadratic variable introduced to define h in the MIP where b_{ij} takes the value h when x_{ij} is 1; 0 otherwise. Competitive-DP is given by,

$$\text{(Competitive-DP) Minimize } \phi_{cmpttv}(x) = h \tag{2.28}$$

$$\text{subject to (2.2)} - \text{(2.9)},$$

$$\sum_{j \in V} (p_j^A + p_j^B) b_{ij} \geq \sum_{j \in V} (p_j^A - p_j^B) x_{ij} \quad \forall\ i \in V, \tag{2.29}$$

$$\sum_{j \in V} (p_j^A + p_j^B) b_{ij} \geq -\sum_{j \in V} (p_j^A - p_j^B) x_{ij} \quad \forall\ i \in V, \tag{2.30}$$

$$b_{ij} \leq x_{ij} \quad \forall\ i, j \in V, \tag{2.31}$$

$$b_{ij} \leq h \quad \forall\ i, j \in V, \tag{2.32}$$

$$b_{ij} \geq h - (1 - x_{ij}) \quad \forall\ i, j \in V, \tag{2.33}$$

$$h, b_{ij} \geq 0 \quad \forall\ i, j \in V. \tag{2.34}$$

The competitiveness objective ϕ_{cmpttv} in (2.28) minimizes the maximum margin h. Constraints (2.29)-(2.30) define $\{b_{ij}\}_{i,j \in V}$ in relation to the margins in each district. Constraints (2.31)-(2.33) linearize the quadratic variables $\{b_{ij}\}_{i,j \in V}$, where $b_{ij} = h\ x_{ij}$ for all $i, j \in V$ using standard linearization techniques (Adams and Sherali, 1990). Constraints (2.34) define the non-negativity of the variables.

Note that optimizing for the competitiveness objective may produce results that are not fair with respect to a *partisan* fairness metric such as the efficiency gap or partisan asymmetry. This is permissible since the idea behind ensuring competitive districts is to prevent incumbent gerrymandering, which is independent of the political affiliation of the winners. Consider an example with district vote-shares in three equipopulous districts exhibiting district vote-shares of $\{0.51, 0.53, .55\}$ for party A. Although this solution is highly inequitable with respect to the political parties (with an efficiency gap of 44%), all three districts are competitive within a 10% margin and can be considered fair in terms of their responsiveness to the voters.

2.4 Solution Method

This section presents a solution approach for solving bi-objective PFDPs. Here, each of the political fairness objectives (i.e., efficiency gap, partisan asymmetry, and competitiveness) is paired with the compactness objective. The goal is to create Pareto-optimal district plans that are compact and are also fair with respect to one of the three political fairness criteria considered in this chapter. Compactness is considered a key criterion since it is generally regarded as an essential requirement in practical district plans (National Conference of State Legislatures, 2020). Since optimizing the compactness objective is NP-complete (Megiddo and Supowit,

1984), each of the bi-objective PFDPs are NP-complete. Hence, an optimization algorithm for districting must address two main challenges. First, the practical instance sizes for PFDPs are typically too large to be solved by an exact method. For example, Wisconsin has $1,409$ census tracts (units) and $3,857$ edges between them, and the MIP (2.2)-(2.9) has 7.5×10^6 variables and 6×10^6 constraints. Second, the PFDPs have multiple conflicting objectives, and the solution approach must produce solutions that highlight the inherent trade-offs between the objectives. This section presents a *multilevel* optimization framework which tackles these challenges.

Algorithm 2.1: Schematic of a multilevel districting algorithm

Input : Level 0 adjacency graph $G_0 = (V_0, E_0)$, number of districts K, population $p_i \ \forall \ i \in V$, number of voters $p_i^A, p_i^B \ \forall \ i \in V$, population deviation threshold $\tau \geq 0$, number of levels L, epsilon-constraint parameter ϵ

Output : z_0^*, the district solution at level 0

$\{G_0, G_1, \ldots, G_L\} \leftarrow$ Coarsen from level 0 through L;

$z_L^* \leftarrow$ Find a Pareto-optimal solution at the L-th level using the $\epsilon-$constraint method;

for *level* $l \in \{L, L-1, \ldots, 1\}$ **do**

  $z_{l-1}^* \leftarrow$ Execute heuristic improvement at G_{l-1} with z_l^* as initial solution;

In this multilevel approach, the problem instance is reduced in size in multiple levels, an exact method solves the reduced instance to (near-)optimality, and the solution is mapped back to the original instance with local search refinements. An outline of the multilevel approach is presented in Algorithm 2.1. There are three stages in the approach: (a) coarsening, (b) solving an exact optimization problem at the coarsest-level, and (c) un-coarsening. The coarsening stage contracts the graph instances in multiple steps, thereby creating levels of graphs that are progressively coarser (i.e., smaller). This is achieved by finding a matching in every graph level and merging every pair of matched units to create a coarsened unit in the next graph level. The number of levels of coarsening is predetermined such that the size of the coarsest level graph is within a desired range. The coarsest level of graph is used as an input to the exact solution stage, where the districting problem is solved to optimality (or near-optimality). In the un-coarsening stage, the optimal solution at the coarsest level is mapped backed to the finer levels, with heuristic local search improvements at each of the levels. See Appendix A.5 for how the multilevel algorithm for a districting problem differs from its application to graph partitioning problems.

The rest of this section details the multilevel algorithm presented in this chapter, organized as follows. Section 2.4.1 discusses the matching-based coarsening scheme used to create a coarse level instance. Section 2.4.2 presents an $\epsilon-$constraint method to find Pareto-optimal solutions at the coarsest level by solving a series of MIPs. Section 2.4.3 discusses the un-coarsening procedure that maps the coarse-level solution back to the original instance level.

2.4.1 Coarsening

The coarsening stage creates multiple levels of progressively smaller graphs by *merging* a subset of units at every level. Starting from the given adjacency graph $G_0 = G$ and number of levels $L \geq 1$, graphs $\{G_l = (V_l, E_l)\}_{l=1}^{L}$ are constructed such that pairs of units in G_{l-1} are merged to form the units in G_l for every $l \in [L]$. When two units are merged, their populations and numbers of voters are added. The pairs of units to be merged are chosen by solving a matching problem. Given an undirected graph $G = (V, E)$, a *matching* is a subset of edges $\mathcal{M} \subseteq E$ such that no two edges share the same node (unit), i.e., $(i, j), (i, j') \in \mathcal{M} \Rightarrow j = j'$. Once a

matching is obtained, the two units at the endpoints of every matched edge are merged to form a single unit in the next level. Note that merging a pair of units is equivalent to imposing a constraint that those two units have to be assigned to the same district in the districting problem solved at the coarse instance.

Based on experimental investigations, it was observed that merging two highly populated units leads to an imbalanced population distribution among the units in the next level, $\{p_i\}_{i \in V_{l+1}}$, thereby making it difficult to find district plans that satisfy the population balance constraints. Hence, the objective of the matching-based coarsening considered in this chapter is to ensure that the highly populous units are not matched together. For each edge $(i, j) \in E$, let $u_{i,j} = (p_i + p_j)$ be the edge weight that captures the total combined population of units i and j. The *minmax matching problem* finds a matching $\mathcal{M} \subset E$ of a particular size such that the maximum weight among the matched edges, $\max_{e \in \mathcal{M}} u_e$, is minimized. Two matching types are considered. A *maximum* (or maximum cardinality) matching is a matching of the largest size. When a maximum matching is used for coarsening, it leads to the most reduction in size of each graph level — by at most half the number of nodes (in which case it is said to be a *perfect* matching). A *maximal* matching $\mathcal{M}$ is a matching such that no additional edges can be added to $\mathcal{M}$ while $\mathcal{M}$ remains a matching, i.e., $\forall\ e \in E \backslash \mathcal{M}$, $\mathcal{M} \cup \{e\}$ is not a matching. On one hand, coarsening using a maximum matching will yield the smallest possible graph in the next level, but may create highly populous units to achieve this reduction. In contrast, since there are at least as many maximal matchings as maximum matchings (i.e., since every maximum matching is also a maximal matching), the minmax maximal matching problem permits a larger set of matchings; hence, its optimal matching may be better (i.e., smaller weight) than that of the minmax maximum matching problem. Therefore, these two matching schemes provide a trade-off between achieving a coarse graph with the fewest units (i.e., maximum matching) and achieving a slightly larger coarse graph that avoids creating highly populous units (i.e., maximal matching). To explore this trade-off, this chapter uses both maximum and maximal matchings to perform the coarsening.

Maximum Matchings

The *minmax maximum matching problem* (MMP), also known as the *bottleneck matching problem* finds a maximum matching such that the weight of the heaviest edge is minimized. MMP can be solved using binary search as presented in Gabow and Tarjan, 1988. The first step is to find the size of a maximum matching in G, which takes $O(n^2 m)$ time using Edmonds' blossom contraction algorithm (Edmonds, 1965), where $n = |V|$ and $m = |E|$. Even though this chapter uses Edmonds' algorithm to find a maximum matching, note that Micali and Vazirani, 1980's augmenting path algorithm is faster and takes $O(\sqrt{n}m)$ time. For the instance of Wisconsin solved in this chapter with $n = 1,409$ and $m = 3,857$, Edmonds' algorithm implemented using Eppstein, 2003 takes less than 5 seconds to find a maximum matching, so its computational requirements are not a limiting factor in the experiments presented in this study.

While Edmonds' algorithm will find a maximum matching for G, it does not necessarily produce a minmax maximum matching, since it does not attempt to minimize the weight of the heaviest edge in this matching. However, Edmonds' algorithms can form the basis of an algorithm to find a minmax maximum matching. Let $l^* \leq |V|/2$ be the size of the maximum matching in G. Next, the edges in G are sorted in the non-decreasing order of the edge weights, $\{e_1, e_2 \ldots, e_m\}$, which takes $O(m \log n)$ time. To find the minmax maximum matching, the idea is to find the smallest subset of the sorted edges $\{e_1, e_2, \ldots, e_t\}$, for $t \in [m]$, such that there exists a maximum matching of size l^* in the subgraph induced by this subset of edges. Note that the maximum edge weight in this subgraph is from edge e_t, and hence finding the smallest subset of edges is equivalent to minimizing the maximum edge weight (i.e., the objective of the MMP). This is achieved using

binary search on $t \in [m]$, which takes $O(\log m)$ steps. Edmonds' algorithm is used as a subroutine to find the maximum matching for each subgraph induced by the subset of edges. Hence, the overall algorithm takes $O(n^2 m \log m)$ time.

Maximal Matchings

The *minmax maximal matching problem* (MLP) finds a maximal matching such that the weight of the heaviest edge is minimized. While MMP can be solved in polynomial time, MLP is NP-complete via a reduction from the classical dominating set problem (Lavrov, 2019). In this chapter, a *greedy* algorithm is used to find a solution to MLP as follows. First, the edges are sorted in a non-decreasing order based on their weights, which takes $\mathcal{O}(m \log n)$ time. Starting with an empty matching $\mathcal{M} = \emptyset$, edges are visited in the sorted order. A visited edge e is added to $\mathcal{M}$ if and only if e is not incident to any edge already in $\mathcal{M}$. This is verified by maintaining a binary array of size $|V|$ (where each element corresponds to a unit in V), that tracks whether each unit currently appears in the matching. Hence, it takes $\mathcal{O}(1)$ time to query the endpoints of an edge, to decide whether to add the edge to M, and to update the array if the edge is added. The algorithm terminates when all the edges in the sorting are visited once, and there are $\mathcal{O}(m)$ steps in total. Overall, this algorithm runs in $\mathcal{O}(m \log n)$ time.

Appendix A.6 discusses the approximability of MLP for the population-based edge weights considered in this chapter assuming that the unit populations are strictly positive. Theorem 2 shows that the objective value of any maximal matching (and in extension, the solution returned by the greedy algorithm) is at the most ρ times the objective value of the optimal maximal matching to MLP, where $\rho \coloneqq \max_{(i,j)\in E} \max\{p_i/p_j, p_j/p_i\}$ is the maximum population ratio among neighboring units in G.

Theorem 2. *Consider a connected graph $G = (V, E)$, positive unit populations $p_i > 0$ for every unit $i \in V$, and edge weights $u_{ij} = p_i + p_j$ for every edge $(i, j) \in E$. Let M^* be an optimal solution to the minmax maximal matching problem with inputs $(G, \{u_{ij}\}_{(i,j)\in E})$, and let $u^* \coloneqq \max_{e\in M^*} u_e$ be its maximum edge weight. Let M be an arbitrary maximal matching with maximum edge weight $u' \coloneqq \max_{e\in M} u_e$. Then, $u'/\rho \leq u^* \leq u'$, where $\rho \coloneqq \max_{(i,j)\in E} \max\{p_i/p_j, p_j/p_i\}$.*

The proof is presented in Appendix A.6. Note that this result does not apply to instances which possess zero-population units. Further, Appendix A.6 also constructs an adversarial example where the greedy algorithm returns the worst possible maximal matching with an objective value that is ρ times the optimal objective value. These results are shown for the population-based edge weights considered in this chapter, where $u_{i,j} = (p_i + p_j)$ is the weight on edge $(i, j) \in E$. Therefore, the approximability of MLP for the general class of edge weights remains an open problem.

Matching-Based Coarsening

Once a matching $\mathcal{M}_l$ is chosen at level $l \in [L]$, the matched edges are merged to obtain a coarse graph. The next graph level $G_{l+1} = (V_{l+1}, E_{l+1})$ is created by merging every matched edge from G_l into a single node in V_{l+1}. Let $\mathcal{C}_l : V_l \to V_{l+1}$ be the mapping of a node from level l to level $l+1$ such that for $i, j \in V_l$, $\mathcal{C}_l(i) = \mathcal{C}_l(j)$ if and only if $(i, j) \in \mathcal{M}_l$, where $\mathcal{M}_l$ is a minmax weight matching in G_l. The matching $\mathcal{M}_l$ is either the maximum matching found using Edmonds' algorithm described in Section 2.4.1, or the maximal matching found using the greedy algorithm described in Section 2.4.1. The edges of the coarse-level graph G_{l+1} are defined such that if and only if there is at least one edge $(i, j) \in E_l \backslash \mathcal{M}_l$ in G_l, the edge $(\mathcal{C}_l(i), \mathcal{C}_l(j)) \in E_{l+1}$

in G_{l+1}. The population of a node in V_{l+1} is the sum of the populations of the corresponding matched units in V_l. The number of voters are also similarly aggregated.

When the maximal/maximum matching is close to a perfect matching, the next level graph has approximately half the number of nodes of the previous level. Hence, the number of levels L can be chosen such that the size of the graph in the coarsest level, $|V_L|$, is within the capacity of an optimization solver to exactly solve the districting problem for the coarsened instance. For example, when starting from a graph of size $|V_0| = 1,409$ (i.e., the number of census tracts in Wisconsin), if an exact solution for PFDP can be obtained for a 200-node graph, assuming near-perfect matchings at each level, L can be set to $\lceil \log_2(\frac{1409}{200}) \rceil = 3$ levels.

2.4.2 Exact Method for PFDP

This subsection presents the inner stage of the multilevel algorithm (line 2 of Algorithm 2.1) comprising an exact solution strategy for solving a bi-objective PFDP. The coarsest level of graph, G_L, is used as the input. In a multi-objective optimization problem with $\mathcal{O} \geq 2$ minimization objective functions given by $\phi = \{\phi_1, \phi_2, \ldots, \phi_{\mathcal{O}}\}$ and a solution space $\mathcal{X}$, two solutions $\hat{\mathbf{x}}, \mathbf{x} \in \mathcal{X}$ can be compared using dominance relationships. The solution $\hat{\mathbf{x}}$ is said to *dominate* $\mathbf{x}$ if $\hat{\mathbf{x}}$ has a strictly better objective value in at least one of the $\mathcal{O}$ objective functions and does not have a worse value in any of the other objective functions, i.e., $\phi_{q'}(\hat{\mathbf{x}}) < \phi_{q'}(\mathbf{x})$ for some $q' \in [\mathcal{O}]$, and $\phi_q(\hat{\mathbf{x}}) \leq \phi_q(\mathbf{x})$ for all $q \in [\mathcal{O}]$. Further, $\hat{\mathbf{x}}$ is said to be *Pareto-optimal* if $\hat{\mathbf{x}}$ is not dominated by any other solution in $\mathcal{X}$. Hence, a Pareto-optimal solution is one that cannot be improved with respect to one objective function without sacrificing the value of another objective function. The goal of solving a multi-objective optimization problem is to find the set of all Pareto-optimal solutions. Let $\mathcal{P}$ denote the set of all Pareto-optimal solutions, or the Pareto set. The *Pareto-frontier* is the set of objective values $\{\{\phi_1(\hat{\mathbf{x}}), \phi_2(\hat{\mathbf{x}}), \ldots, \phi_{\mathcal{O}}(\hat{\mathbf{x}})\}\}_{\hat{\mathbf{x}} \in \mathcal{P}}$ in the objective space corresponding to Pareto-optimal solutions. The set of values in the Pareto-frontier provides a trade-off of the extent to which one objective function can be made better by sacrificing another objective function.

The $\epsilon-$constraint method is a popular method to compute the Pareto set in multi-objective problems (Chiandussi et al., 2012). This method first selects one of the objectives as a *primary* objective, while the other *secondary* objectives are introduced as constraints of the form $\phi_q(\mathbf{x}) \leq \epsilon_q \ \forall \ \mathbf{x} \in \mathcal{X}$, where ϵ_q is a user-imposed upper bound on a secondary objective $q \in [\mathcal{O}]$. This is an iterative procedure where the values of ϵ_q are gradually reduced, starting from a large value. The amount by which the values of ϵ_q are reduced in every iteration should be small enough that no non-dominated solution is skipped over. For each value of ϵ_q, a single-objective problem is solved with the primary objective function as the main objective. The optimal solution found in each iteration is stored, and is a candidate to be a Pareto-optimal solution. The method terminates when the value of ϵ_q is so small that the optimization solver either returns an infeasibility certificate or does not find a feasible solution within the time limit. If the primary objective function is such that several solutions could have the same value, the $\epsilon-$constraint method may produce dominated solutions. When the dominated solutions are removed from the set of candidates, the rest of the solutions form the Pareto set.

For a bi-objective PFDP like those that will be solved in Section 2.5, the compactness objective (ϕ_{comp}) as defined in (2.10) is chosen as the primary objective since the moment of inertia function (i.e., sum of the weighted squared distances from the district centers) is structurally similar to the $p-$median problem, which is known to possess good linear programming lower bounds (Salazar-Aguilar et al., 2011). Hence, solving a single-objective problem in each iteration of the $\epsilon-$constraint method is relatively more computationally efficient with compactness as the primary objective. This was experimentally verified by setting each of

the other fairness objectives as the primary objective, which resulted in a lack of convergence to optimality. Additionally, since compactness is universally regarded as an essential requirement in political districting, it is considered a primary criterion (National Conference of State Legislatures, 2020). Hence, the solution approach is to solve Compact-DP supplemented by constraints related to the other three fairness metrics (as formulated in their single-objective problems) and the $\epsilon-$constraints. The corresponding $\epsilon-$constraints for ϕ_{EG}, ϕ_{cmpttv} and ϕ_{PA} are given by constraints (2.35), (2.36) and (2.37), where ϵ_{EG}, ϵ_{PA} and ϵ_{cmpttv} are upper bounds on the efficiency gap, partisan asymmetry, and competitiveness, respectively.

$$-\epsilon_{EG}\sum_{i\in V}(p_i^A+p_i^B)\leq\sum_{i\in V}a_i\leq\epsilon_{EG}\sum_{i\in V}(p_i^A+p_i^B), \tag{2.35}$$

$$\sum_{k=1}^{K}|\hat{\omega}_k-(1-\hat{\omega}_{K-k+1})|\leq K\epsilon_{PA}, \tag{2.36}$$

$$h\leq\epsilon_{cmpttv}. \tag{2.37}$$

As described in Section 2.3, a_i is the difference in wasted votes between parties A and B in a district with center $i\in V$, $\hat{\omega}_k$ is the minimum average vote-share required for party A to win $k\in[K]$ districts, and h is the maximum margin across all the districts. Note that the absolute value in constraint (2.36) can be linearized by standard linearization techniques. In particular, for each $k\in[K]$, the absolute value term in constraint (2.36) is replaced by an additional continuous variable $\Omega_k\geq 0$, along with additional constraints $-\Omega_k\leq(\hat{\omega}_k-(1-\hat{\omega}_{K-k+1}))\leq\Omega_k$. Then, the constraint $\sum_{k=1}^{K}\Omega_k\leq K\epsilon_{PA}$ implies constraint (2.36).

In every iteration of the $\epsilon-$constraint method, it is important to decrement the value of ϵ such that no non-dominated solutions is skipped between successive iterations. In this chapter, ϵ is decremented by 10^{-7} from the corresponding fairness value of the solution found in the previous iteration. This value is obtained by first expressing each of the three fairness objectives (ϕ_{EG}, ϕ_{cmpttv} and ϕ_{PA}) as fractions, and noting that for the Wisconsin instance solved in this chapter, 10^7 is larger than the denominator term in each of these fractions. Hence, when ϵ is decremented by 10^{-7}, no non-dominated solutions is skipped between successive iterations.

While this section describes a method to obtain a Pareto-frontier, the case study presented in this chapter finds an *approximate Pareto-frontier*. The approximation arises from three reasons. First, in a particular iteration for a given ϵ, if the optimization solver terminates due to time limit, it may return a non-optimal solution which is not Pareto-optimal. Second, in this scenario, it is possible that non-dominated solutions in future iterations may be skipped over. Third, in the last iteration, when the solver terminates due to time limit without a feasible solution, it is still theoretically possible that a non-dominated solution exists whose fairness value is smaller than the (smallest) ϵ. To minimize these issues in the case study, the time limit for each iteration of the $\epsilon-$constraint method is generously set to 24 hours.

2.4.3 Un-coarsening

The final stage of the multilevel algorithm (lines 3-5 in Algorithm 2.1) iteratively maps the optimal solution from level L back to level 0. Note that the optimal solution at level L may not be optimal at lower levels, and hence this solution could potentially be improved using a heuristic. Let $z_{l+1}:V_{l+1}\rightarrow[K]$ be the district solution at level $l+1$. First, an initial district solution at level l is created, i.e., $z_l(i)=z_{l+1}(\mathcal{C}_l(i))$ for all $i\in V_l$, where $\mathcal{C}_l$ is the mapping function defined in the coarsening procedure. Then, using the local search method described in Appendix A.6.1, this solution is improved further with respect to ϕ_{comp} while enforcing

contiguity, population balance and an $\epsilon-$constraint. After L levels of un-coarsening, the algorithm returns the locally-optimized solution z_L.

2.5 Case Study in Wisconsin

This section applies the solution approach described in Section 2.4 to approximate the trade-offs between compactness and each of the political fairness metrics (i.e., efficiency gap, partisan asymmetry, and competitiveness) for congressional districting in the state of Wisconsin. The population data and adjacency information are obtained from the U.S. Census Bureau, 2010, and are processed using QGIS. Wisconsin has $|V| = 1,409$ census tracts (units) and $K = 8$ congressional districts with ideal district population $\overline{P} = 710,873$. Although census blocks are more granular than census tracts, this chapter uses census tracts as the basic units since fewer than 5% of the census tracts are split between multiple districts according to Wisconsin's 2010 district plan. Voter information is compiled as the average of the 2012 and 2016 presidential election results (McGovern, 2017; The Guardian, 2018). The Democratic (D) and Republican (R) parties have an approximately equal overall vote-share (51.6% and 48.4% respectively) across the state. This information is available at the precinct level, which is aggregated to the county level, and then distributed to the census tracts proportionally based on their populations, after which the number of voters in each census tract is rounded down to the nearest integer. While more sophisticated aggregation techniques exist, where precinct-level data is de-aggregated to census blocks and is then aggregated to census tracts (MGGG, 2021), the coarse approximation of voting patterns yielded by disaggregating from counties still provides an example instance on which to test the multilevel algorithm for the PFDP proposed in this study.

Population balance requires that the relative difference between every district's population and the ideal district population lies within a given deviation threshold $\tau \geq 0$ (Kalcsics and Ríos-Mercado, 2019). Alternatively, it is also common to require that the relative difference between the most and least populous districts be within a tolerable range (Grofman and Cervas, 2020). Even though the Supreme Court does not set an exact legal limit for the value of τ, they require that district populations be "as nearly [equal] as is practicable" (Wesberry et al. v. Sanders, Governor of Georgia, et al., 1964). In the case study presented in this chapter, τ is set to 2%, which allows for a maximum deviation of up to 14,217 people in each district from the ideal district population. To achieve the goal of analyzing the trade-offs between the fairness objectives, a 2% deviation is preferred since it allows the algorithm to examine the tradeoffs within a larger pool of candidate district plans compared to using a smaller deviation threshold. Moreover, even though a district plan with a 2% deviation may be considered unconstitutional, such a plan can be refined at the census block level to achieve a plan with a tighter population deviation. DeFord et al., 2021b note that a plan with a 1% deviation is "easily tuneable to 1-person deviation by refinement at the [census] block level", and the same refinement technique can be extended to a plan with a 2% deviation. Further, the 2% deviation has precedence in prior work in Congressional districting, such as in DeFord and Duchin, 2019. As a proof of concept, Section 2.5.2 presents district plans with τ as low as 0.25% to showcase the ability of the multilevel algorithm to produce practical district plans even without census block level refinements.

The computations reported in this chapter are performed using a 2.7 GHz Intel Core i5 machine with 8 GB memory. The algorithms are written in Python 3.7. The computational times are reported using both clock time (seconds) and CPU time (ticks). For each of the PFDPs, the initial solution from the multistart local search is first computed with the termination criteria being either 100 iterations of the multistart local search, or a time limit of 60 minutes. To solve the MIP formulations, IBM's CPLEX Optimizer 12.6 is used

with its default settings which include a host of cuts and heuristics. The `MIPGap` parameter is set to zero in order to find the optimal solutions to the MIPs. The initial solution found using the multi-start heuristic is passed into the root of the branch-and-cut tree. A time limit of 24 hours is set for each run of solving the MIPs. If the optimality gap is non-zero, the best feasible solution is returned as an approximate optimal solution. If no feasible solution is found, the bi-objective multilevel algorithm terminates, and the current Pareto frontier is returned.

The rest of this section is organized as follows. Section 2.5.1 presents results from the two matching-based coarsening procedures used to coarsen Wisconsin's census tract level instance. Section 2.5.2 presents district plans produced from solving the Compact Districting Problem with the population deviation threshold (τ) value ranging from 0.25% to 2%. Sections 2.5.3, 2.5.4 and 2.5.5 present district plans obtained from solving three bi-objective problems, each with compactness as the primary objective and a respective political fairness metric — efficiency gap, partisan asymmetry and competitiveness — as the secondary objective. The approximate Pareto-frontiers produced highlight the trade-offs between compactness and the three political fairness metrics.

2.5.1 Coarsening

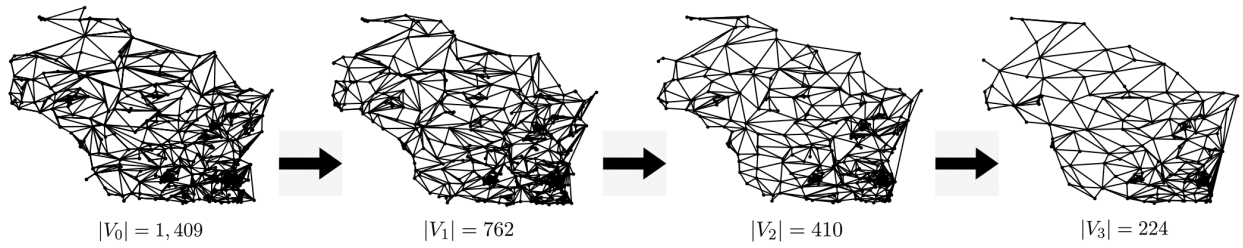
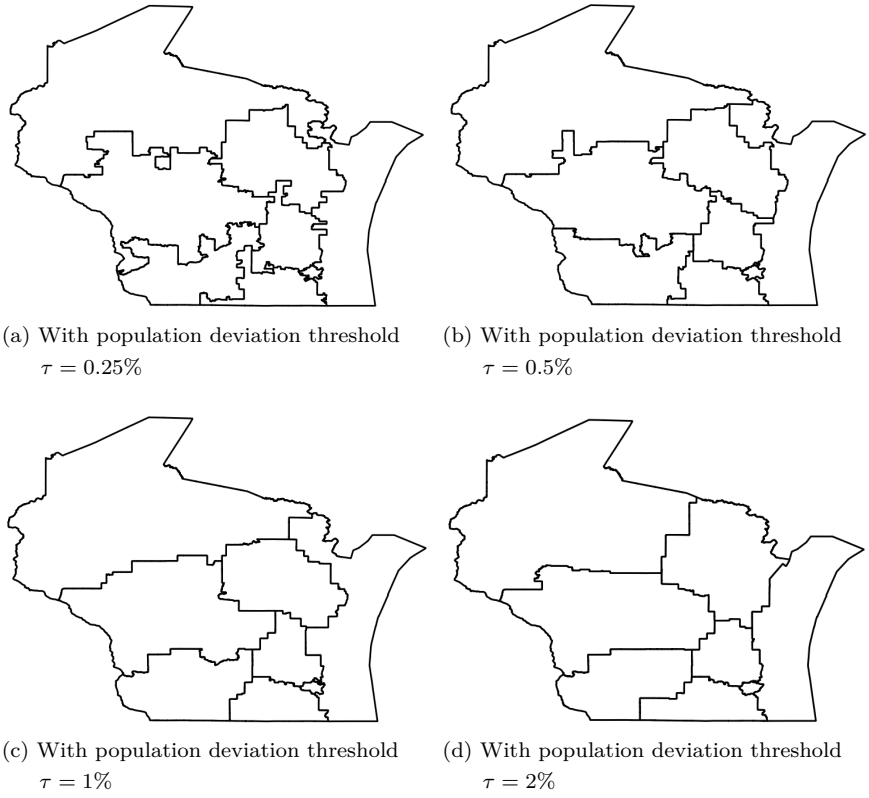


Figure 2.3: Adjacency graphs of Wisconsin for three levels of coarsening. Each level is obtained by merging matched units in a minmax maximal matching in the previous level, as described in Section 2.4.1.

Starting with the adjacency graph at the finest level (census tracts), the matching-based coarsening procedure proceeds as described in Section 2.4.1. Two coarse-level instances are produced when coarsening using (a) a minmax maximum matching (MM), and (b) a minmax maximal matching (ML). For both methods, the number of levels L is set to three, since it was observed that the optimization solver performs well for instance sizes around 200 units, and three levels of near-perfect matchings would attain instances near this size.

Coarsening using MM reduces the graph size to $|V_3| = 177$ nodes, whereas ML reduces the graph size to $|V_3| = 224$ nodes. Figure 2.3 depicts the adjacency graphs of the three levels of graphs produced when coarsening using ML. As expected, coarsening using MM produces a smaller graph instance than when using ML, and hence could lead to faster computation of Pareto-optimal solutions at the coarsest level. However, in the coarsest graph G_3, the population of the most-populous unit when using MM is $52,207$, whereas when using ML is $47,932$. The subsequent Sections 2.5.3 — 2.5.5 present results for PFDPs using both coarsening methods. Note that the definition of compactness depends on the structure of a graph instance. Since the two coarsening methods produce two different coarse graph instances, the compactness values cannot be directly compared between the two methods. Hence, in Figures 2.5, 2.7 and 2.9, the vertical axes representing compactness have different scales between the two coarsening methods.

2.5.2 Compact Districts



(a) With population deviation threshold $\tau = 0.25\%$

(b) With population deviation threshold $\tau = 0.5\%$

(c) With population deviation threshold $\tau = 1\%$

(d) With population deviation threshold $\tau = 2\%$

Figure 2.4 Compact district plans for varying tightness of the population balance constraints.

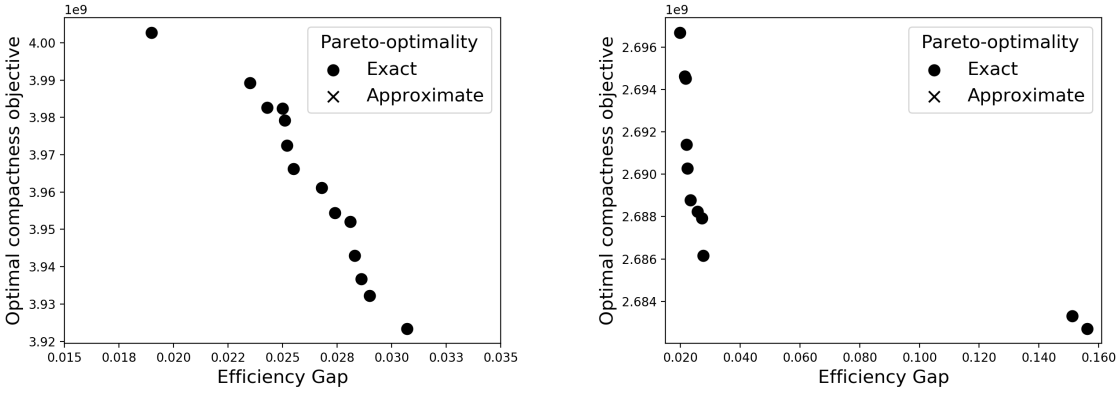
The multilevel algorithm first solves the traditional Compact-DP formulated by (2.2)-(2.10). Coarsening using ML produces the graph instance $G_3 = (V_3, E_3)$ after three levels of coarsening. Since this section serves as a proof-of-concept that the multilevel approach can produce compact districts meeting tighter population balance tolerances than $\tau = 2\%$, coarsening using MM is not presented here as results using ML are sufficient to demonstrate this capability. In the definition of compactness in (2.10), the distance d_{ij} between every pair of units $i, j \in V_3$ is measured by the number of edges in the shortest path between i and j in G_3. Note that Hess et al., 1965's original model defines d_{ij} to be a general distance function. Mehrotra et al., 1998 conclude that the shortest path-based definition used in this chapter produces less geographical bias than using Euclidean distances since it is invariant of the geographical sizes of the units. This definition of compactness is used in the rest of Section 2.5.

To observe the impact of the population balance threshold (τ) on the compactness of the district plan produced by the algorithm, four values for τ are chosen from $\{0.25\%, 0.5\%, 1\%, 2\%\}$. Figure 2.4 depicts the four district plans produced for these values of τ. The inner-level exact optimization stage solves the coarsened instance with 224 units within 1 hour for each of the four settings of τ.

Further, the plan obtained with $\tau = 0.25\%$ is testament to the ability of the multilevel algorithm to produce district plans that satisfy practical population balance requirements. It can be seen that as τ is increased from 0.25% to 2%, the corresponding districts produced by the algorithm are visually more compact. Comparing the four plans, the majority of the units in the districts remain in the same district across the

plans. The differences arise in the district boundaries, where the compact shapes are perturbed to satisfy a tighter population balance constraint for smaller values of τ. These plans illustrate the trade-off between the compactness objective and the tightness of the population balance constraint. To allow more flexibility when approximating the trade-offs between compactness and the three political fairness objectives, a population deviation threshold of 2% is set in the rest of this section.

2.5.3 Efficient Districts

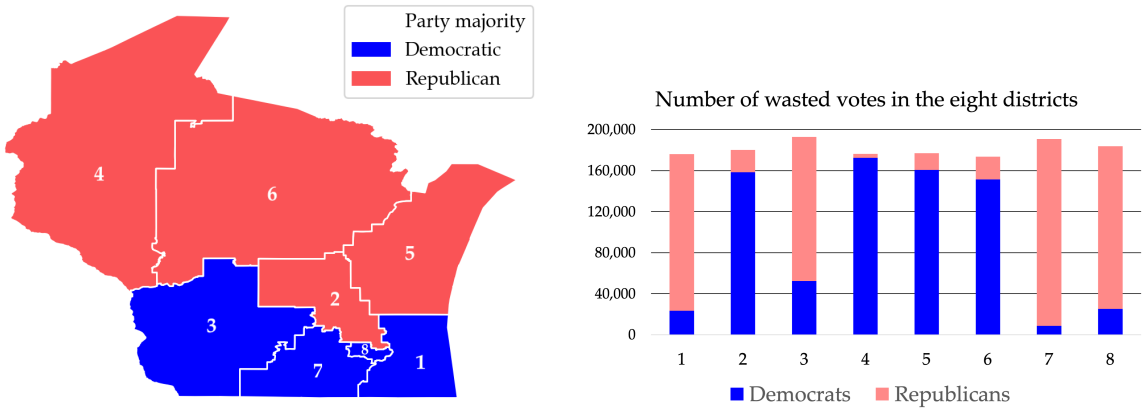


(a) When coarsening using *maximum* matchings

(b) When coarsening using *maximal* matchings

Figure 2.5: Approximate Pareto-frontier using the coarsest graph instance G_3 highlighting the trade-off between the efficiency gap and compactness objectives. Each solution represents either an exact or an approximate solution found in the coarsest graph instance G_3.

To approximate the trade-off between compactness and efficiency gap (EG), the approximate Pareto-optimal solutions are obtained by solving the PFDP with objective set $\{\phi_{comp}, \phi_{EG}\}$, formulated by (2.2)-(2.10), (2.14)-(2.20) and (2.35) using the coarsest levels obtained from the two coarsening methods. The value of ϵ_{EG} is iteratively reduced from 0.5 (i.e., the theoretical maximum possible value for EG), and the MIP is solved either to optimality or until the time limit is reached. Figure 2.5 depicts the Pareto-frontiers highlighting the trade-off between the compactness and EG objectives. Coarsening using MM and ML produce 14 and 11 district plans respectively, with similar median EG values of 0.023 and 0.027, respectively. Every iteration terminated to optimality within the 24 hour time limit. The computational results are presented in Section A.7.1 of the appendix. All the solutions terminated to optimality.



(a) Party majorities in the eight districts classified based on their margins of victory.

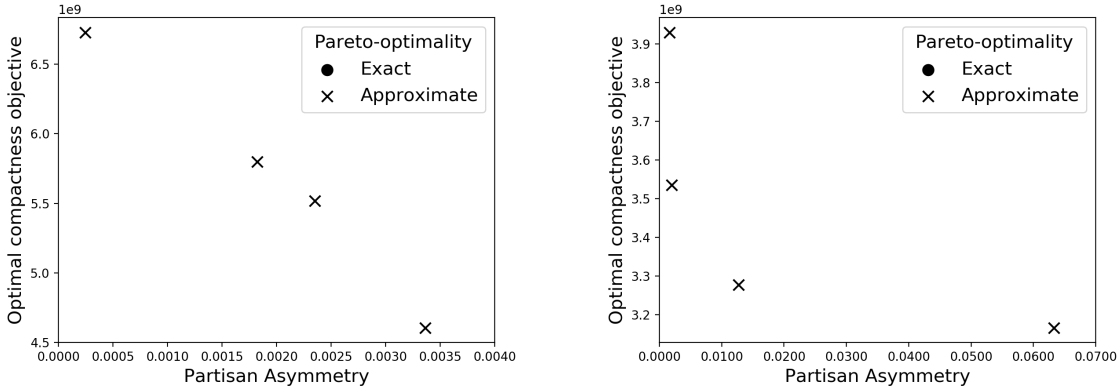
(b) Number of votes wasted by both parties.

Figure 2.6 An efficient district plan for Wisconsin with an efficiency gap of 0.0188.

When the solutions reported in Table A.7.1 are un-coarsened with local search improvements to compactness, the smallest value of EG attained is 0.0188 (0.0198) when the coarsening procedure uses MM (ML). The district plan corresponding to an EG of 0.0188 is depicted in Figure 2.6a. Both the parties win four districts each, and hence the seat-share is proportional to the overall vote-share. Figure 2.6b depicts the number of votes wasted by the two parties in the eight districts. This district plan is an ideal candidate to a districting process that seeks to minimize the difference in wasted votes between the two parties. Even though this district plan minimizes EG, this plan is neither competitive nor symmetric. It can be seen all eight districts are highly packed and cracked by both parties to an almost equal degree, leading to poor competitiveness as measured by a maximum margin of 27.19%. Moreover, this plan also has a high partisan asymmetry value of 0.0218, which is much larger than the best asymmetry value of 0.0002 achieved in Section 2.5.4, due to a large difference in the two parties' vote-seat curves.

2.5.4 Symmetric Districts

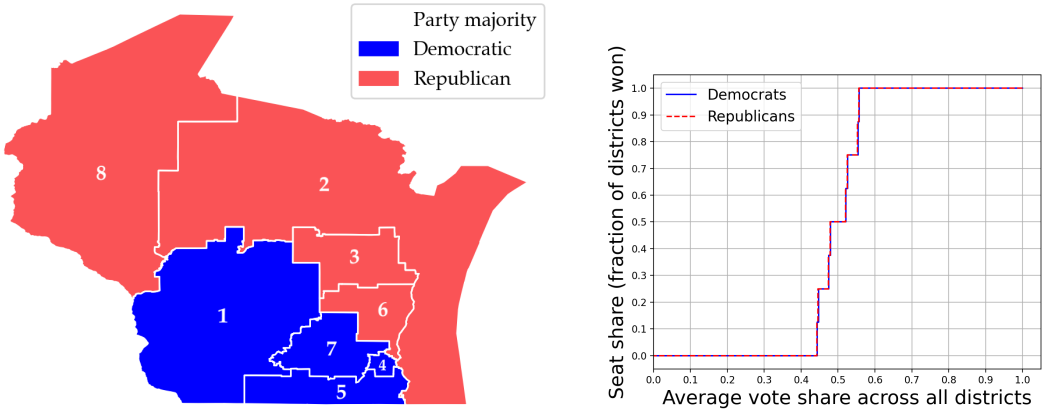
To approximate the trade-off between compactness and partisan asymmetry (PA), the approximate Pareto-optimal solutions are obtained by solving the PFDP with objective set $\{\phi_{comp}, \phi_{PA}\}$, formulated by (2.2)-(2.10), (2.36) and the linearized constraints (A.2)-(A.14), (A.17)-(A.22) given in the proof of Proposition 1 in the appendix A.4. A reduced time limit of 6 hours is used for each run of solving the MIP since the 24 hour time limit produced large branch-and-bound trees whose sizes exceeded the computer's memory. This is due to the relative complexity of the MIP that models PA. To produce the district plans, the value of ϵ_{PA} is iteratively reduced from 1 (i.e., the theoretical maximum possible value for PA). The solver terminates with an approximate solution in every iteration owing to the complexity of solving the MIP. Coarsening using MM and ML produce 4 district plans each, with median PA values of 0.0021 and 0.0073, respectively. Figure 2.7 depicts the approximate Pareto-frontiers highlighting the trade-off between the compactness and PA objectives. The computational results for the eight district plans produced are presented in Section A.7.2 in the appendix.



(a) When coarsening using *maximum* matchings.

(b) When coarsening using *maximal* matchings.

Figure 2.7: Approximate Pareto-frontier using the coarsest graph instance G_3 highlighting the trade-off between the partisan asymmetry and compactness objectives. Each solution represents either an exact or an approximate solution found in the coarsest graph instance G_3.



(a) Party majorities in the eight districts classified based on their margins of victory.

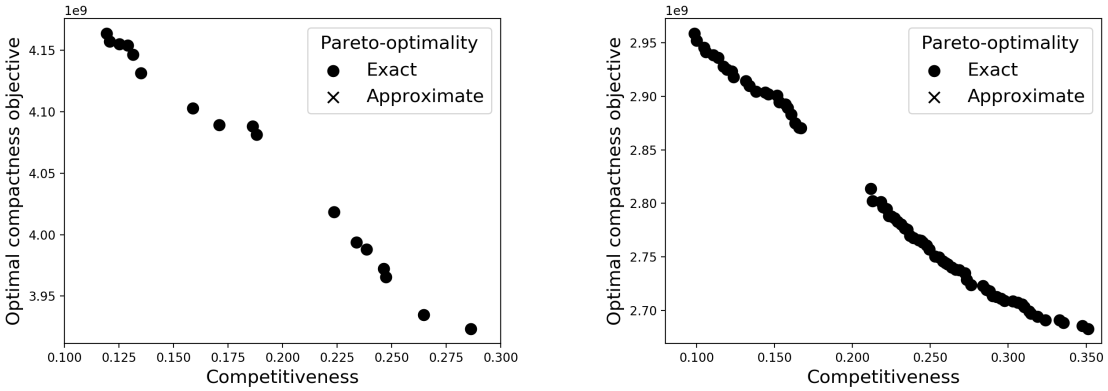
(b) Vote-seat curves for both parties.

Figure 2.8 A symmetric district plan for Wisconsin with a partisan asymmetry of 0.0002.

When the solutions reported in Table A.7.2 are un-coarsened with local search improvements to compactness, the smallest value of PA attained is 0.0002 (0.0016) when the coarsening procedure uses MM (ML). Further, note that among the six district plans with PA values less than 0.01, five of them have R winning 4 districts out of 8 and the sixth one has R winning 5 districts out of 8. This suggests that the majority of the symmetric district plans align with vote-seat proportionality. Figure 2.8a depicts a district plan with a PA value of 0.0002 found when coarsening using MM. In this plan, both the parties win four districts each. Figure 2.8b shows the corresponding vote-seat curves for the two parties. It can be seen that the two curves are close to each other with an area of 0.0002 between them, signifying that both parties gain or lose seats symmetrically when voting levels fluctuate from their actual values. Hence, this district plan is an ideal candidate for a districting process that seeks to minimize PA. Even though this plan has been optimized for PA, it is not competitive. The high concentration of D voters in district 4 gives this plan a competitiveness value (maximum margin) of 14.39%. However, since both parties win an equal number of

districts, the number of seats won by each party is proportional to its overall vote-share, giving rise to a small EG value of 0.0216 where D wastes more voters than R.

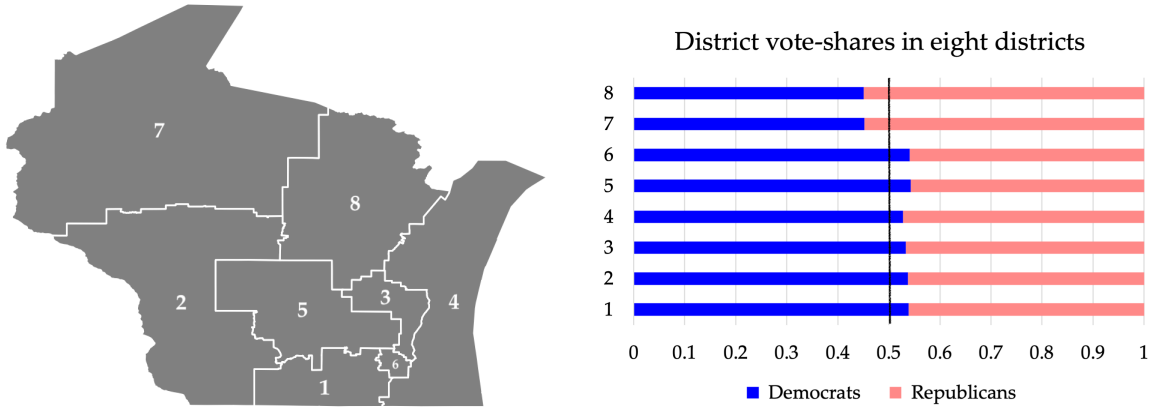
2.5.5 Competitive Districts



(a) When coarsening using *maximum* matchings.

(b) When coarsening using *maximal* matchings.

Figure 2.9: Approximate Pareto-frontier using the coarsest graph instance G_3 highlighting the trade-off between the competitiveness and compactness objectives. Each solution represents either an exact or an approximate solution found in the coarsest graph instance G_3.



(a) The eight districts classified based on their margins of victory.

(b) District vote-shares for both parties.

Figure 2.10 A competitive district plan for Wisconsin with a maximum margin of 9.78%.

To approximate the trade-off between compactness and competitiveness, the approximate Pareto-optimal solutions are obtained by solving the PFDP with objective set $\{\phi_{comp}, \phi_{cmpttv}\}$, formulated by (2.2)-(2.10), (2.29)-(2.34), and (2.37). The value of ϵ_{cmpttv} is iteratively reduced from 1 (i.e., the theoretical maximum possible value for the maximum margin), and the MIP is solved either to optimality or until the time limit is reached. Figure 2.9 depicts the Pareto-frontiers highlighting the trade-off between the compactness and competitiveness objectives. Coarsening using MM and ML produce 17 and 72 district plans respectively,

with respective median competitiveness values of 18.6% and 23.8%. Every iteration terminated to optimality within the 24 hour time limit. The instances produced by MM are solved slightly slower (with an average of 64.2 minutes) than the instances produced by ML (with an average of 59.1 minutes).

When the solutions reported in Table A.7.3 are un-coarsened with local search improvements to compactness, the smallest value of competitiveness value attained is 11.94% (9.78%) when the coarsening procedure uses MM (ML). Figure 2.10a depicts a district plan with all eight districts have their margins less than 9.78% found when coarsening using ML, and Figure 2.10b depicts the corresponding district vote-shares. Party D disproportionally wins six out of the eight districts, and hence this plan has a high EG value of 0.2202. This plan also has a high PA value of 0.0374 (compared to the district plan in Section 2.5.4 with a PA of 0.0002) due to the asymmetry in the seat gains between the two parties. The computational details are presented in Section A.7.3 of the appendix.

2.6 Conclusions

This chapter addresses the issue of gerrymandering by providing a practical multi-objective framework that can be adopted by a districting process that explicitly focuses on political fairness. The mathematical models presented in this chapter optimize a set of fairness metrics that cater to different facets of fairness. The computational experiments demonstrate an algorithm for creating district plans using two objectives at a time. The case study for congressional districting in Wisconsin highlights that solutions that are approximately optimal with respect to one objective may be poor with respect to another. These results reiterate the need for a holistic multi-objective perspective in the design of political districts to best serve the interests of multiple stakeholders.

This chapter re-frames the districting process and serves as a starting point to investigate how political fairness can be integrated into optimization-based approaches for district design. There are several aspects of this framework that can be improved with future work. First, optimizing over one pair of fairness objectives could unintentionally produce solutions with poor values for the other objectives, as was observed in the Wisconsin case study reported in Section 2.5. Hence, extending the $\epsilon-$constraint method to optimize more than two objectives at a time could yield a more holistic approach that considers simultaneous trade-offs between multiple objectives. Second, the algorithm relies on the use of historic population and voting data to draw districts for the future. Incorporating potential variations in voting trends into the mathematical models by using robust optimization could produce district plans that are more relevant to future needs when populations and voter preferences are unknown. Third, improving the efficiency of exactly solving the PFDPs using methods that utilize the structure of the MIPs (e.g., using decomposition techniques) could yield a more exhaustive Pareto-frontier within a shorter computational time.

Chapter 3

A Case Study on Optimization for Political Redistricting in Practice

3.1 Introduction

Political redistricting is the process of redrawing district boundaries for congressional and state legislative elections every 10 years, and it is subject to various legal criteria outlined in the U.S. and state Constitutions. For example, Arizona's Constitution outlines a wide range of criteria that span geographical considerations such as contiguity, compactness and preservation of political subdivisions, demographic considerations such as population balance and majority-minority districts, and partisan concerns such as political competitiveness. An algorithm that draws district maps must cater to these subjective and often conflicting legal criteria.

Optimization algorithms are well suited to serve the needs of political redistricting in practice. When legal criteria for redistricting are quantifiable, the redistricting process can be modeled as an optimization problem by identifying specific criteria as objective(s) to optimize, while treating other criteria as constraints. For example, ensuring that all districts are contiguous can be naturally modeled as a constraint, whereas subjective measures such as compactness or competitiveness are more aptly modeled as objectives. Furthermore, the legal language used in constitutional requirements for redistricting indirectly solicits an optimization approach. For example, five of the six criteria in Arizona's Constitution must be favored "to the extent practicable." An optimization algorithm aims to do precisely that—optimize given objective(s) to the *extent practicable*, governed by constraints and computational limits.

Despite the potential benefits of using an optimization algorithm for redistricting, significant challenges must be overcome for practical adoption. The first challenge is the **complexity of modeling** the redistricting process. Practical redistricting often requires the consideration of a wide range of criteria, such as the criteria specified in the Arizona Constitution. Existing optimization methods have primarily focused on a subset of criteria, such as population balance, contiguity, and compactness. However, capturing all of the legal criteria outlined in a state's Constitution has proven to be a significant challenge, partly due to the subjectivity of these criteria and the lack of transparent data. For example, it is common to require that districts preserve "communities of interest," but it is often unclear which communities the map drawers have chosen to preserve by the end of the process. The second challenge is **computational intractability.** Many optimization methods cannot scale to the large input sizes needed for practical redistricting. The number of census blocks can range from thousands to nearly one million, as in the case of Texas. The third challenge is the **alignment**

with a practical redistricting procedure. For advanced technology to assist with practical adoption, it must be transparent and capable of integrating with established practices. For example, Arizona uses a multi-stage process where a district map undergoes a series of modifications while incorporating public feedback. However, mathematical optimization methods such as Mixed Integer Programming can be opaque to non-experts in the field, which could impede their implementation in practice.

This chapter presents a case study in Arizona, introducing a local search-based optimization framework for redistricting its nine congressional districts. This framework incorporates various existing methods, including a multilevel algorithm that transitions between various levels of graph data and a local search method that iteratively improves selected objectives. We tailor this framework for Arizona with three guiding principles: (i) to satisfy all the legal criteria specified in the U.S. and Arizona Constitutions, (ii) to be computationally scalable to large input sizes, and (iii) to exhibit a clear alignment with the existing redistricting process in Arizona. A framework that adheres to these principles will likely be more valuable to a practitioner as a tool to draw legally viable district maps. In this chapter, we use Arizona as a case study to demonstrate the effectiveness of this framework. However, the algorithm design process can be extended to other states.

The well-defined constitutional guidelines in Arizona and the transparent redistricting process instituted by the Arizona Independent Redistricting Commission (AIRC) make Arizona an ideal state for this case study in applying an optimization method. The framework presented in this chapter incorporates all six legal criteria outlined in Arizona's Constitution. It heuristically optimizes *compactness*, *competitiveness* and population balance, while treating the rest as constraints. Compactness measures the geographical shape of districts, and rewards visually well-rounded districts. A competitive district map ensures that districts are not unreasonably packed with one party's voters, which is a gerrymandering tactic. The AIRC defines a district as competitive if its *vote spread* (i.e., the fractional difference between the two major parties' voters in a district) is below 7% (AIRC, 2021a). Given the numerous criteria tackled in this framework, the AIRC's feedback during the drafting stages of the 2021 redistricting cycle gave us insights into effectively operationalizing these criteria. For example, we ensure that district populations are *balanced* to the maximal extent, i.e., are as close to a one-person deviation from the ideal (average) district population as possible.

Further, the presented method mimics the AIRC's iterative process, where an initial district map undergoes refinements across several phases. For context, during the AIRC's drafting stages of the enacted map for the 118^{th} Congress (CD118), the fifth stage improves compactness and competitiveness, while other stages catered to criteria such as preserving communities of interest and creating majority-Hispanic districts. Our optimization algorithm operates in four stages: stage 1 creates an initial district map, stages 2 and 3 heuristically optimize compactness and competitiveness, respectively, and stage 4 heuristically optimizes population balance. This decomposed approach effectively addresses the often conflicting criteria in redistricting. Additionally, the algorithm's computational tractability stems from its ability to traverse multiple layers of granularity, such as census tracts, block groups, and blocks, allowing the map to evolve throughout the stages.

This chapter aims to validate the effectiveness of optimization algorithms in political redistricting, particularly in handling conflicting objectives and large input sizes. We benchmark our approach against the CD118 baseline by generating a set of legally valid district maps. The key question addressed in this chapter is: Can an optimization method create legally valid maps with better compactness and competitiveness than CD118 without compromising the other legal criteria? To foster transparency and accessibility, we have developed and released a user-friendly interface to interact with the first three stages of the algorithm, available at: https://redistricting.cs.illinois.edu/#optimap.

The rest of the chapter is organized as follows. Section 3.2 reviews existing literature on redistricting criteria and optimization algorithms that cater these criteria. Section 3.3 discusses background information on Arizona redistricting, the datasets used in this chapter, the pre-processing stage used in this chapter. Section 3.4 outlines the optimization algorithm used to create the district maps presented in this case study. Section 3.5 presents an empirical analysis of the algorithm's performance and a curated collection of district maps. Section 3.6 comments on the generalizability of the framework to other states and the limitations that should be considered when interpreting the results of the framework. Section 5.7 provides conclusions and future research directions.

3.2 Background

Political redistricting as an optimization problem has been well-studied. This section provides an overview of existing literature that motivates and forms the basis for our case study. Redistricting typically requires the consideration of multiple criteria. Section 3.2.1 details the legal and political background on common redistricting criteria. Section 3.2.2 examines classical optimization models and methods that address fundamental redistricting criteria such as compactness, contiguity, and population balance. Section 3.2.3 reviews literature focusing on specific non-political criteria (i.e., that do not require voting data), such as the preservation of communities of interest and the incorporation of majority-minority districts. Section 3.2.4 discusses approaches targeting political fairness criteria, such as competitiveness, efficiency gap, and partisan asymmetry. Finally, Section 3.2.5 discusses two commonly employed methodological frameworks in redistricting and graph partitioning algorithms: the multilevel algorithm and local search. These frameworks serve as the foundation for our case study.

3.2.1 Redistricting Criteria

This section provides an overview of key existing literature concerning common redistricting criteria. The specific criteria covered include compactness in Section 3.2.1, population balance in Section 3.2.1, majority-minority representation in Section 3.2.1, and political fairness metrics in Section 3.2.1. This review provides a legal background on quantifying these criteria essential for a practical map drawing algorithm.

Compactness

Intuitively, a district is *compact* if its boundary resembles a simple shape, such as a circle, rectangle, or regular polygon. A non-compact district (i.e., one with a convoluted shape) not only suggests nefarious intent but also makes representation more confusing for the representative and their constituents (Grofman and Cervas, 2021). Judges and political scientists consider compactness an important guard against gerrymandering (Kaufman et al., 2021), and 18 states, including Arizona, impose compactness requirements for congressional districts (Ballotpedia, 2020b).

While compactness is widely acknowledged as an important redistricting criterion, there is no universally accepted definition for it, though many mathematical definitions have been proposed (Grofman and Cervas, 2021). There are generally three standard approaches to measuring compactness: using perimeters (i.e., boundary lengths), areas, and dispersion (i.e., the spread of geographical units relative to a district's geographical center). In its simplest form, the perimeter metric quantifies the total perimeter lengths of all the districts (ESRI, 2021b); smaller is preferred. A widely used perimeter-based metric is the *Polsby-Popper*

metric, calculated as the ratio between a district's area and the area of a circle with the same perimeter length (Polsby and Popper, 1991). Alternatively, the *Reock* metric, a popular area-based metric, is calculated as the ratio between a district's area and the area of the circle that circumscribes it (Reock, 1961). In both these metrics, a ratio close to 1 suggests a circle-like shape. Moreover, Duchin and Tenner, 2018 recently introduced a graph-theoretic metric for district compactness, which counts the number of census blocks in the perimeter and divides it by the number of blocks in the district, weighted by population. Note that this metric can be viewed as a discrete analog of the perimeter metric (Clelland et al., 2021). Besides using perimeter and area, the dispersion-based *moment-of-inertia* metric calculates the weighted distances of geographical units from a center (Fan et al., 2015). This metric is popular among exact optimization models, which will be discussed in Section 3.2.2. Survey results indicate that none of the currently available definitions fully align with human perceptions of compactness (Kaufman et al., 2021), yet they remain valuable for quantifying compactness in map drawing algorithms.

Population Balance

The landmark U.S. Supreme Court cases of the 1960s, including Baker v. Carr, 1962, Wesberry et al. v. Sanders, Governor of Georgia, et al., 1964 and Reynolds v. Sims, 1964, established the "one person, one vote" principle, meaning congressional districts within a state should have approximately the same populations to provide equal representation to all constituents (McGhee, 2020). Imbalance in district population is typically measured by their deviation from the average (ideal) district population. It is generally considered impractical to achieve near-perfect population balance in a map drawing algorithm because doing so would require dealing with large input sizes, which are typically intractable (Ricca et al., 2013; Validi et al., 2022). However, to guarantee the legal validity of a congressional district map, it is necessary to achieve a population deviation as close as possible to just one person.

Minority Representation

The Voting Rights Act (VRA) of 1965 aims to prevent racial discrimination in voting. According to the U.S. Supreme Court's interpretation in Thornburg v. Gingles, 1986, this act supports the establishment of majority-minority districts when there is a significant concentration of a minority group (Niemi et al., 1990). A *majority-minority district*, referred to as a *VRA district*, is commonly defined as one where a minority group or a combination of minority groups make up a simple majority of the district's population (Ballotpedia, 2021a). The 118th Congress is said to be the most racially and ethnically diverse in history, with 136 among the 435 congressional districts being majority-minority (Ballotpedia, 2021a).

The threshold for considering a district majority-minority varies based on the state and specific minority. Goedert et al., 2023 adopted a 40-45% threshold range for Black-majority districts in southern states. This choice stems from estimating the probability of electing a minority representative for a given minority population percentage. For example, a district with 60% Black population in Maryland has a 95% likelihood of electing a Black representative, in contrast to 91% in Alabama. Yet, Goedert et al., 2023 categorize districts with over 50% Black Voting Age Population as "Black-majority" districts. In a different context, Becker et al., 2021 applies a 60% threshold in Texas. Given the distinct requirements of each state regarding minority populations, a map drawing algorithm should allow the map drawer to adjust the threshold as needed.

Competitiveness and Partisan Fairness

Fairness in redistricting, while subjective, can be quantified using specific metrics that represent various perspectives of fairness. There are three common notions of fairness: competitiveness, proportionality and symmetry. Competitiveness focuses on the voters, ensuring that political representatives remain responsive. The latter two address fairness concerning political parties and quantify the extent that one party might have an advantage over another. These metrics have been predominantly used in assessing gerrymandering (e.g., outlier analysis using ensemble methods), and have rarely been used in active map-creation.

Ensuring that districts are competitive can help reverse the effect of any potential incumbent gerrymandering, where "safe seats" are created to protect incumbent candidates. For example, competitive districts increase voter turnout and responsiveness to constituents (Gordon and Huber, 2007; Goedert et al., 2023; Divounguy et al., 2023). A recent study on election results in North Carolina found that a competitive 55-45 district increases turnout by 1% per election compared to an uncompetitive 80-20 district (Ainsworth et al., 2022). In practice, five states have emphasized competitiveness in their state constitutions (National Conference of State Legislatures, 2020). Furthermore, competitiveness is one among three criteria used by The Gerrymandering Project, a consortium that uses mathematical tools to diagnose unfair outcomes in district maps (Gerrymandering Project, 2023). Therefore, ensuring that districts are competitive is important for promoting fairness through accountability and responsiveness.

Beyond competitiveness, political scientists and redistricting reformers often focus on partisan metrics such as the efficiency gap (EG) and partisan asymmetry (PA). EG measures the difference between the two parties' *wasted votes* divided by the total number of votes; wasted votes for a particular party comprise those cast either (a) in a district where the party is expected to lose, or (b) above the 50% of votes needed to win in a district where the party is expected to win (Stephanopoulos and McGhee, 2015). EG quantifies imbalances in how many of each party's voters are packed into districts where their candidate wins by a landslide and/or cracked across districts where their candidate loses narrowly. Based on a historical analysis of EG in U.S. district maps, its creators suggest an 8% threshold as reasonable regarding partisan fairness (Stephanopoulos and McGhee, 2015). While there is academic debate on the merits of the efficiency gap (Chambers et al., 2017; Stephanopoulos and McGhee, 2018), it has been used in the courts to overturn gerrymandered maps (Bernstein and Duchin, 2017).

While EG captures proportionality, PA measures symmetry. PA compares each party's seat share (i.e., the fraction of districts in which that party has a majority of votes) under hypothetical shifts in their vote-share (i.e., a party's share of voters) across all districts (Grofman and King, 2007). Such counterfactual analysis produces a vote-seat curve for each party, denoted as a function that maps each vote-share to its corresponding seat share over the range of hypothetical shifts. The *asymmetry* of the district map is then calculated as the area between the two parties' vote-seat curves (Grofman, 1983; Katz et al., 2020). Though fraught with computational issues and paradoxes (DeFord et al., 2021b), partisan symmetry has earned moderate endorsements from judges (see LULAC v. Perry, 2006).

3.2.2 Optimization Approaches with Canonical Considerations

Optimization models and methods for political redistricting have early origins, such as a heuristic method introduced by Vickrey, 1961 and a formal integer programming model developed by Hess et al., 1965. Subsequently, numerous methods have emerged in academic literature, spanning exact methods (Garfinkel and Nemhauser, 1970; Plane, 1982; Mehrotra et al., 1998; Shirabe, 2009; Oehrlein and Haunert, 2017; Validi

et al., 2022) to heuristic methods such as local search (Ricca and Simeone, 2008), network flows (George et al., 1997), and simulated annealing (D'Amico et al., 2002). Exact methods focus on finding globally optimal solutions. However, their applicability is limited to problems with specific structures, limiting their effectiveness to smaller input sizes. Conversely, heuristic methods can accommodate multiple criteria and solve larger practical input sizes, albeit at the cost of not guaranteeing globally optimal solutions.

A predominant theme among existing optimization approaches is their focus on a limited set of three canonical criteria: *contiguity*, *compactness*, and *population balance* (Ricca and Scozzari, 2020). Of these, contiguity and compactness focus on a district's shape. Contiguity is computationally challenging to implement, and some methods relax contiguity requirements in their solution phase (Gentry et al., 2015). Recent advances have improved contiguity enforcement in both exact and heuristic methods. The contiguity-enforcement methods by Oehrlein and Haunert, 2017 and Validi et al., 2022 solve a compactness-seeking objective measured by the sum of distances from the center of each district. King et al., 2018 provided an efficient method to verify contiguity in a local search method.

Compactness is often considered an objective in optimization approaches. While there are numerous definitions of compactness as outlined in Section 3.2.1, existing optimization models predominantly optimize the moment-of-inertia metric (Hess et al., 1965; Mehrotra et al., 1998; Validi et al., 2022). This preference is due to other metrics like Polsby-Popper being non-linear, thereby increasing both mathematical and computational complexity. Recent studies by Belotti and Buchanan, 2023 and Fravel et al., 2023 delve into the intricacies of optimizing non-linear compactness measures. Despite the popularity of moment-of-inertia in optimization models, it is rarely used in practice. Besides moment-of-inertia, the perimeter metric is linear, allowing for efficient computation within improvement heuristics. Minimizing the perimeter metric is a weighted generalization of minimizing the number of cut edges, a problem studied by Validi and Buchanan, 2022. Furthermore, the perimeter metric is one of the eight scores that the AIRC employs to assess Arizona's district maps during the map drawing process (AIRC, 2021a). Hence, an optimization algorithm that optimizes the perimeter metric is expected to be both computationally efficient and yield compact district maps.

Population balance has been modeled as a constraint (Hess et al., 1965) or an objective (Levin and Friedler, 2019; Olson, 2013). When modeled as a constraint, existing methods typically set a large threshold for the population deviation from the average, such as 0.5%. However, congressional redistricting requires deviations to be as close to one person as possible. A key challenge for an optimization method to achieve such a small population deviation is constructing districts from census blocks, which leads to a very large optimization problem over the most finely-grained census geography; existing optimization methods do not scale to this level. To the best of our knowledge, Levin and Friedler, 2019 is the only optimization method that has achieved a population balance of *one person* in 42 out of 43 states, using a divide-and-conquer method, but ignores compactness and all other criteria except contiguity. To create district maps that can be adopted in practice, an optimization algorithm must achieve a population deviation close to one person while simultaneously achieving other required criteria.

Despite these advancements in computational methods, the problem of partitioning a graph into population-balanced districts is NP-Complete (Garey et al., 1976). Finding a balanced, contiguous, and optimally compact district map is intractable for large input sizes. For example, it can take more than three days to optimally solve an instance with $1,409$ units using the cutting planes method proposed by Validi et al., 2022. In comparison, Arizona has $155,444$ census blocks. Hence, there is a need for scalable methods that can handle large practical instance sizes and incorporate the numerous legal criteria encountered in practical

redistricting scenarios.

3.2.3 Optimization Approaches Tailored for Non-Political Criteria

While optimization approaches have traditionally focused on canonical considerations, a practical algorithm must incorporate other redistricting criteria. One such criterion is the preservation of communities of interest (COIs) and political subdivisions like counties, townships, and municipalities. Preserving these subdivisions ensures that residents with similar needs and interests are considered when making policy decisions. However, these criteria have been seldom considered in optimization methods since there is "no global consensus on their legitimacy" (Ricca and Scozzari, 2020). Birge, 1983 is the only known optimization approach that models preserving political subdivisions such as counties.

As discussed in Section 3.2.1, incorporating a certain number of majority-minority districts is mandated by the U.S. Constitution to ensure adequate representation for voters belonging to a racial minority population (Aleinikoff and Issacharoff, 1993). Although there are numerous studies on this criterion's political and socio-economical implications (Epstein and O'Halloran, 1999), existing optimization methods treat this criterion as an afterthought or disregard it (Ricca and Scozzari, 2020). To the best of our knowledge, Cannon et al., 2023 is the only method that explicitly incorporates majority-minority districts; this approach proposes a random walk that restarts from a district map with the most majority-minority districts. However, this approach does not accommodate multiple criteria, and whether it can be extended to fulfill the various criteria outlined in Arizona's Constitution remains unclear.

3.2.4 Optimization Approaches Incorporating Partisan Fairness

Political fairness metrics (such as those discussed in Section 3.2.1) have recently been incorporated into optimization methods (Swamy et al., 2023; Gurnee and Shmoys, 2021; Liu et al., 2020). These methods explicitly optimize fairness metrics such as the efficiency gap and competitiveness. However, existing works in this area serve as proofs-of-concept when optimizing fairness metrics, and these studies ignore other considerations, such as ensuring a strict population balance, incorporating communities of interest and majority-minority districts.

3.2.5 General Algorithmic Frameworks

This section examines two algorithmic frameworks, the multilevel algorithm and local search, which underpin the optimization framework presented in this case study.

Multilevel Algorithm

A multilevel algorithm is a widely used framework for solving graph partitioning problems on large graphs (Karypis and Kumar, 1998). It comprises three phases: (i) a *coarsening* phase that reduces the input graph size to a small-enough size via multiple levels of graph contractions, (ii) an *optimization* phase that finds an optimal or approximately optimal solution in the smaller graph, and (iii) an *uncoarsening* phase that transforms the solution back to the original large graph, adding heuristic local refinements. The flexibility of creating a solution in multiple granularities makes this method effective in large-scale graph partitioning. Given the connection between graph partitioning and districting, this method has recently been adopted to political redistricting (Swamy et al., 2023). Furthermore, Magleby and Mosesson, 2018 explored a stochastic

variant of the multilevel algorithm to simulate random district maps using random coarsening choices. Hence, an optimization method adopting a multilevel algorithm could benefit from its scalability.

Local Search

A local search methods have been well-adapted to political redistricting (Ricca and Simeone, 2008; King et al., 2012), since they are scalable to large input sizes and are intuitive to explain to non-experts in optimization. Local search makes incremental alterations to a district map to progressively improve a chosen objective function. A key local search parameter is the termination criterion. A common criterion is to terminate after a pre-defined maximum number of consecutive iterations without any improvement in the objective (Avci and Topaloglu, 2015).

A local search neighborhood defines the allowed alterations. For political redistricting, two types of local search neighborhoods are common. Given a district map, a *Flip* iteration re-assigns one geographical unit from its current district to another neighboring district (Ricca and Simeone, 2008). Each Flip iteration makes a small change to a district map. A more expansive neighborhood can be defined by completely redrawing large sections of the district boundaries. As a special case, DeFord et al., 2021a designed a spanning-tree method to redraw the boundary between two neighboring districts, called *recombination* or *Recom*. Even though Recom is designed in the context of a Markov chain Monte Carlo (MCMC) method, Recom can accelerate a local search method by making drastic changes to the district boundaries.

3.3 Arizona Redistricting Preliminaries

This section outlines the mathematical, legal, and political considerations in designing our optimization framework for redistricting in Arizona. To formalize the inputs, Section 3.3.1 introduces the mathematical notation used in this chapter. Section 3.3.2 outlines the data from Census 2020 and past election results necessary to create maps. Section 3.3.3 discusses Arizona's enacted map CD118 and its limitations. Section 3.3.4 lists the redistricting requirements specified in Arizona's Constitution. Finally, Section 3.3.5 describes a pre-processing stage that prepares the inputs for our algorithm.

3.3.1 Mathematical Notation

Redistricting data can be encoded in a graph data structure as follows. Let $G = (V, E)$ be a graph, where the node set V is the set of geographical *units* (such as census tracts, census block groups, or census blocks) and an edge exists in E between two units if they share a non-zero boundary length between them. The population, voters and racial demographics are available for each unit. For unit $i \in V$, let p_i be its total population, and let p_i^A and p_i^B be the number of voters for parties A and B (such as the Democratic and Republican parties), respectively. Furthermore, let $p_i^{\min}$ be the number of residents in i who identify either as a single racial/ethnic minority or as any racial/ethnic minority, depending on the state's definition. For example, for Arizona's 2020 congressional redistricting process, $p_i^{\min}$ counts the number of residents who identify as Hispanic. Moreover, we also have geographical boundary information between adjacent units. Between every pair of adjacent units $(i, j) \in E$, let $b_{i,j} > 0$ denote the shared boundary length. These data will be used to measure the six redistricting criteria in Arizona.

Redistricting is the process of creating a *district map*, which assigns the units to districts. We can describe a district map as a partitioning of the set of units V into K districts (or parts), where K is a predetermined

parameter. For example, Arizona has $K = 9$ congressional districts. To represent a partitioning of V, let $\{V_k\}_{k=1}^K$ denote the collection of K subsets where each subset V_k is the set of units assigned to district $k = 1, 2, \ldots, K$; these sets are mutually exclusive (i.e., $V_k \cap V_{k'} = \emptyset$ for all distinct $k, k' = 1, 2, \ldots, K$) and collectively exhaustive (i.e., $\cup_{k=1}^K V_k = V$). We can also equivalently represent a district map by the mapping of units to districts. For each unit $i \in V$, let $z(i) = 1, 2, \ldots, K$ denote the district to which unit i is assigned (i.e., $z(i) = k$ if and only if $i \in V_k$). The rest of this chapter represents a district map by either representation ($\{V_k\}$ or z) based on convenience and clarity.

3.3.2 Redistricting Data

Redistricting requires data describing a state's geography, population, racial/ethnic demographics, and election results. We compile these data from publicly available sources such as the U.S. Census Bureau, 2020, OpenPrecincts, 2020, and the AIRC's redistricting hub (AIRC, 2021b). This section describes the data required for Arizona redistricting.

Spatial Data

Districts comprise geographic units such as census blocks, census block groups, census tracts, and counties. The 2020 U.S. Census provides spatial data for these units as shapefiles (U.S. Census Bureau, 2020). With spatial data, we determine which units are adjacent (to enforce district contiguity) and compute the length of shared borders between adjacent units (to calculate district perimeters as a measure of compactness). We process the adjacency information from the shapefiles using the free and open-source QGIS software, and the publicly available Shapely Python package. In the 2020 U.S. Census, Arizona's 15 counties contain 1,765 census tracts (see Figure 3.1a), 4,773 census block groups, and 155,444 census blocks, which are the four granularities of building blocks used to create the district maps.

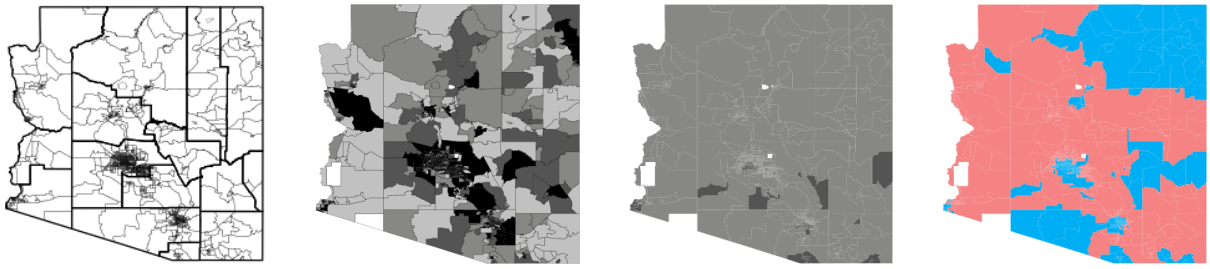
Population and Racial Demographics

In addition to spatial data, the U.S. Census provides population and racial/ethnic demographics for all geographical units (U.S. Census Bureau, 2020). Our algorithm uses population data to ensure that all districts have approximately equal populations. Figure 3.1b depicts the geographical distribution of the population in Arizona. Dividing Arizona's total population of 7,151,502 into nine districts gives an ideal congressional district population of 794,611.

The racial/ethnic demographic data are necessary to construct majority-minority districts as required by the Voting Rights Act (VRA). In Arizona, 30.7% of the state population identifies as Hispanic, and Figure 3.1c depicts the census tracts with Hispanic-majority populations. These data can be used within an optimization algorithm to create the number of Hispanic-majority districts required by the VRA.

Electoral Demographics

Voting data from past elections are used to measure political metrics such as competitiveness, as discussed in Section 3.3.4. Mirroring the AIRC's process, we aggregate the results from nine elections: 2018 President and U.S. Senate; 2020 U.S. Senate, Governor, Secretary of State, Attorney General, State Treasurer, Superintendent of Public Instruction, and State Mine Inspector. Precinct-level voting data for these elections are available from OpenPrecincts, 2020. We only use voting data for the two major political parties, as the AIRC prescribes.



(a) Census tracts (thin lines) and counties (thick lines).

(b) Population heatmap from Census 2020.

(c) Census tracts with a majority Hispanic population (in dark gray) from Census 2020.

(d) Party affiliation: Democratic majorities in blue (darker) and Republican majorities in red (lighter).

Figure 3.1: Spatial, population, racial and electoral demographics in Arizona. The party affiliation in each census tract is obtained by aggregating nine specific election results in 2018 and 2020. The census tracts in white contain zero population.

Since our algorithm creates district maps using census tracts, census block groups, and census blocks, precinct voting data need to be redistributed to those geographical units. Since census blocks are nested inside voting precincts, we first disaggregate the precinct data to census blocks by distributing the voters in each precinct proportionally to the populations of the census blocks in the precinct. Subsequently, the census block-level data are aggregated to the census block groups and census tracts using the hierarchical nesting of these levels.

With these voting data, we can visualize the spatial distribution of Arizona's voters. Based on the nine focus elections, Arizona voters are 48.3% Democratic and 51.7% Republican. Figure 3.1d illustrates the distribution of Democratic and Republican voters in the state.

3.3.3 Arizona's Enacted Congressional District Map, CD118

In November 2000, Arizona amended its state Constitution to create the Arizona Independent Redistricting Commission (AIRC), tasked to redraw congressional and state legislative districts following each decennial census (AIRC, 2021a). The AIRC has emphasized transparency in the redistricting process by actively engaging with the public and demonstrating openness by welcoming draft maps from a wide range of stakeholders. This level of public engagement encourages the public to create high-quality maps using optimization methods such as the framework presented in this case study.

Following the 2020 Census, the AIRC released Arizona's nine congressional districts (labeled CD118), depicted in Figure 3.2, which were subsequently enacted in December 2021 to be served until 2032. Note that CD118 was not explicitly drawn using an optimization algorithm. CD118 has been criticized for having only three competitive districts (Morgan and Howard, 2022; Wong, 2022), and scores poorly in terms of geographical compactness (when scored using the popular perimeter metric), despite Arizona's Constitution emphasizing compactness and competitiveness as two of the six criteria. CD118 has drawn criticism for disproportionately favoring the Republican party (Duda, 2021; Arizona Daily Star, 2021), resulting in four Republican-leaning districts and only two Democratic-leaning districts, despite both parties having a roughly equal share of voters in the state (based on the nine elections in 2018 and 2020 prescribed by the AIRC). Given the potential for CD118 to be improved, the optimization method presented in this chapter uses CD118

as a baseline for creating legally valid district maps.

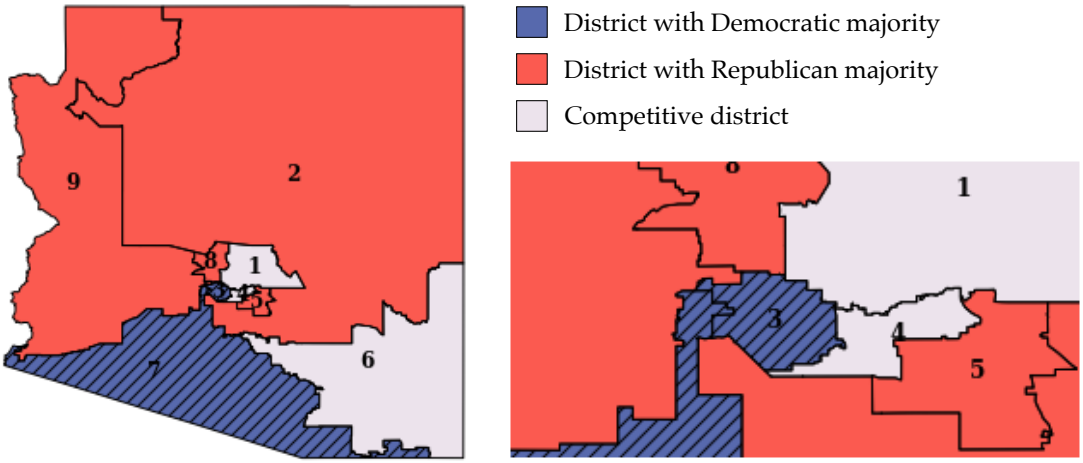


Figure 3.2: Arizona's enacted congressional district map (CD118) for the 118th congressional session (2022-24); all nine districts (left) and districts in the city of Phoenix (right). The two majority-Hispanic districts are shaded.

3.3.4 Arizona's Constitutional Requirements

As amended by Proposition 106, the Arizona Constitution (Arizona, 2021) describes six specific redistricting requirements which guide the design of our framework. This section lists excerpts of relevant text from Proposition 106.

(A) *"Districts shall comply with the United States Constitution and the United States voting rights act;"* (Arizona, 2021)

As discussed in Section 3.2.1, this requirement ensures a certain number of majority-minority districts. Data from the 2020 Census reveal that 30.7% of Arizona's population identifies as Hispanic, and the enacted 2021 district map (CD118) ensures that two out of the nine districts have a Hispanic majority. Hence, a practical map drawing algorithm for Arizona must ensure two Hispanic-majority districts.

Mathematically, a district $k = 1, 2, \ldots, K$ is a *majority-minority district* if the minority population fraction is more than a given threshold, $\alpha_{\text{VRA}} \in [0, 1]$, of the total population, i.e., $\sum_{i \in V_k} p_i^{\text{min}} > \alpha_{\text{VRA}} \sum_{i \in V_k} p_i$. Let $Q \subseteq \{1, 2, \ldots, K\}$ be the majority-minority districts. This requirement ensures the number of such districts meets a minimum, N_{VRA}, decided during the redistricting process. For Arizona, as decided in the process of creating CD118, we require two Hispanic-majority districts, i.e., $N_{\text{VRA}} = 2$. Furthermore, we assume a minority fractional threshold of $\alpha_{\text{VRA}} = 0.5$, consistent with standard legal definitions (Ballotpedia, 2021a).

(B) *"Congressional districts shall have equal population to the extent practicable, and state legislative districts shall have equal population to the extent practicable;"* (Arizona, 2021)

As discussed in Section 3.2.1, this requirement ensures that district populations are as close as possible. Mathematically, let $P_k = \sum_{i \in V_k} p_i$ be the total population in district $k = 1, 2, \ldots, K$. When

seeking an equal population, the ideal district population is the rounded average population given by $\overline{P} = \lceil (\sum_{k=1}^{K} P_k)/K \rfloor$. The deviation from this ideal district population is typically measured by the maximum deviation in district populations among all districts, given by $\max_{k=1,2,\ldots,K} |P_k - \overline{P}|$. CD118 exhibits a maximum deviation of 1 person. It is also common to measure population deviation as a percentage deviation from $\overline{P}$. This deviation for a given map z is given by,

$$\text{Population deviation } \phi_{bal}(z) = \max_{k=1,2,\ldots,K} \frac{|P_k - \overline{P}|}{\overline{P}}. \tag{3.1}$$

For a given deviation threshold $\tau \in [0, 1]$, we call a district map τ%-*balanced* if $\phi_{bal} \leq \tau$.

(C) *"Districts shall be geographically compact and contiguous to the extent practicable;"* (Arizona, 2021)

A popular measure of compactness is the perimeter metric (Young, 1988), calculated by the sum of shared district perimeters (reported in km); this sum excludes perimeter segments that coincide with Arizona's state boundary. This metric is one of the eight compactness scores used by the AIRC (AIRC, 2021a). Other factors being equal, a district with an irregular shape has a larger perimeter than a district with a simpler shape. Hence, minimizing the sum-of-perimeters metric yields more compact districts. We measure compactness of a given map z using the total shared perimeter between districts, given by,

$$\text{Compactness score } \phi_{\text{compact}}(z) = \sum_{(i,j) \in E : z(i) \neq z(j)} b_{i,j}. \tag{3.2}$$

The *contiguity* requirement seeks each district to be a geographically connected region. In a contiguous district map, one must be able to travel from any unit (such as a census tract) in a district to any other unit within the same district by moving through a sequence of adjacent units. Every census tract and census block in Arizona is itself contiguous, so this requirement guarantees contiguous districts. Contiguity can be enforced using the graph representation by requiring the subgraph induced by each district to be *connected*. Let $G[V_k]$ denote the subgraph of G induced by district $k = 1, 2, \ldots, K$. Then, $G[V_k]$ is *connected* (and the district is *contiguous*) if between each pair of vertices $i, j \in V_k$, there is a path entirely in $G[V_k]$. In the district maps produced by our algorithm, we require that all districts are contiguous.

(D) *"District boundaries shall respect communities of interest to the extent practicable;"* (Arizona, 2021)

For the 2022 redistricting process, the AIRC created a portal for the public to submit communities they wish to preserve (AIRC, 2021b). The AIRC's portal announced that they received 910 submissions ranging from very large regions (such as shapes that cover the whole state) to very small (such as shapes that cover a single house). Based on these submissions, the AIRC classified each region based on the number of other submitted regions overlapping with that region, with each region being classified as 'Highest', 'High', 'Medium', 'Low', and 'Lowest'.

A higher overlap count indicates that more submissions want that region preserved. At the end of the map drawing process, the enacted district map CD118 preserves a subset of 18 of these communities. To ensure compliance with this requirement at a similar level as CD118, a map drawing algorithm must also prioritize the preservation of these regions. Section 3.3.5 describes the pre-processing stage for an algorithm to preserve these communities by treating these regions as indivisible units.

(E) *"To the extent practicable, district lines shall use visible geographic features, city, town and county boundaries, and undivided census tracts;"* (Arizona, 2021)

CD118 preserves several administrative regions, including cities, towns, and counties. A pre-processing method described in Section 3.3.5 highlights the administrative regions preserved in CD118. Similar to preserving COIs, the pre-processing stage in Section 3.3.5 preserves the same cities, townships and counties as in CD118. To limit the number of divided census tracts, an optimization algorithm could construct the map first from census tracts, then divide the census tracts into census blocks only as needed to further reduce the population deviation.

(F) *"To the extent practicable, competitive districts should be favored where to do so would create no significant detriment to the other goals."* (Arizona, 2021)

Competitive districts reduce district packing, and improve candidate responsiveness to their voters (Hirano and Snyder, 2012; McCarty et al., 2009; Tapp, 2019). As with compactness, competitiveness can be defined in many ways (DeFord et al., 2020). As a measure of competitiveness, the AIRC reports each district's *vote spread*, the percentage difference between the Democratic and Republican votes within a district based on recent elections, describing districts with a vote spread less than 4% as "highly competitive" and between 4% and 7% as "competitive" (Arizona, 2021). In each district $k = 1, 2, \ldots, K$, the vote spread is the fractional difference in the number of voters for the two parties, given by $v_k = |\sum_{i \in V_k}(p_i^A - p_i^B)| / \sum_{i \in V_k}(p_i^A + p_i^B)$. A smaller vote spread indicates a more competitive district. A map drawing algorithm that produces as many competitive districts as possible ensures compliance with this requirement.

We construct a composite *competitiveness score* that incorporates two measures. The goal is to maximize the number of competitive districts, and among the solutions with the most competitive districts, minimize the maximum vote spread among all districts in order to limit the "packing" of voters. To achieve these dual objectives, we construct a competitiveness score for a given map z given by,

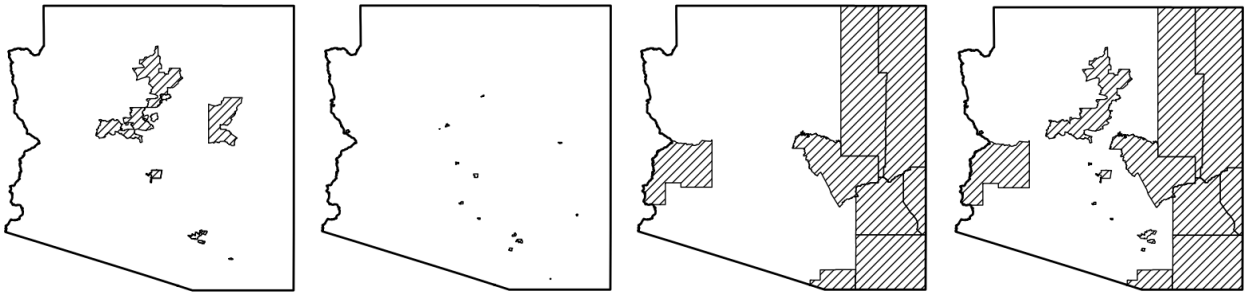
$$\text{Competitiveness score } \phi_{\text{cmpttv}}(z) = \sum_{k=1}^{K} \mathbb{I}\{v_k < 0.07\} - \max_{k=1,2,\ldots,K} v_k, \tag{3.3}$$

where $\mathbb{I}\{.\}$ is the binary indicator function for whether district k is competitive. The first part of the score measures the number of competitive districts and the second part measures the maximum vote spread. We assume that $v_k \neq 1$, since having a district composed entirely of a single party's voters could only happen if there are enough units composed exclusively of a single party's voters to constitute a district, which is not observed in Arizona's 2020 Census data. Since $v_k < 1$, maximizing the competitiveness score first maximizes the number of competitive districts (as in the first term), and then minimizes the maximum vote spread among all districts (as in the second term) lexicographically. Maximizing competitiveness creates greater accountability of candidates to their voters (Friedman and Holden, 2009).

3.3.5 Data Pre-processing

Requirements D and E in the Arizona Constitution require preserving certain regions such as communities of interest (COIs), cities, towns, counties, and census tracts. To enable this preservation, we pre-process the data to treat these regions as indivisible units. We now describe how these regions are collectively preserved.

First, we visualize the COIs derived from AIRC, 2021b. From the 902 submissions that the AIRC received, 18 communities (see Figure 3.3a) with at least a 'Medium' priority were preserved in the enacted map (CD118).



(a) 18 preserved communities of interest.

(b) 33 preserved *places* (such as municipalities, cities, towns, villages and boroughs).

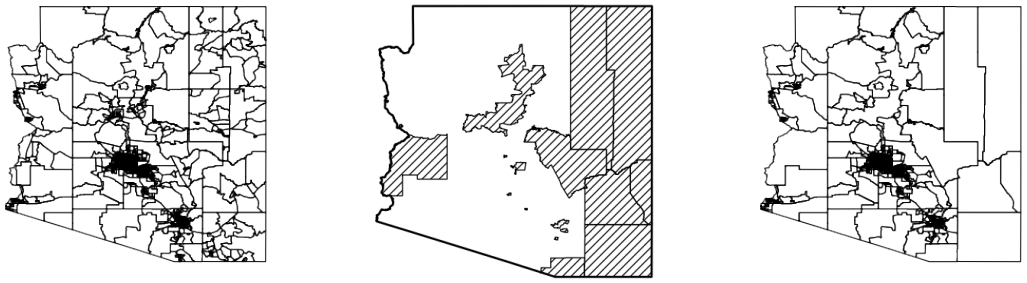
(c) Eight preserved counties.

(d) 19 non-overlapping regions to be preserved.

Figure 3.3 Creating non-overlapping regions to be preserved.

Furthermore, requirement E in the Arizona Constitution requires preserving visible geographical features such as cities, towns, county boundaries, and census tracts. The Census 2020 data have a category called "places," which includes cities and townships. The enacted map CD118 preserves 33 cities and townships, depicted in Figure 3.3b. Furthermore, CD118 preserves eight out of the 15 counties, depicted in Figure 3.3c.

Aggregating all of these regions to be preserved, many of the preserved regions overlap with each other. When two preserved regions overlap, the entirety of both regions has to be assigned to the same district. Based on this principle, we create 19 non-overlapping regions to be preserved, as depicted in Figure 3.3d.



(a) Census tracts of Arizona.

(b) 19 preserved regions.

(c) A mixture of preserved regions and census tracts in the rest of the state.

Figure 3.4: Creating a hybrid census tract graph for Arizona based on census tracts, preserved counties, and communities of interest.

In a graph algorithm for drawing district maps, creating a single node for a preserved region is equivalent to assigning that geographical region to the same district. Hence, we create a "hybrid census tract graph" (as depicted in Figure 3.4) whose nodes are a mixture of the preserved regions and the census tracts in the rest of the state. This graph will be referred to as the hybrid census tract (HCT) graph in the rest of this section. Similarly, we create hybrid census block (HCB) and hybrid census block group (HCBG) graphs.

3.4 Optimization Method

The proposed optimization framework aims to provide a practical tool for drawing district maps in Arizona. The design of this algorithm follows three guiding principles. First, the district map produced by the algorithm should be legally valid for adoption in practice. We have ensured that the algorithm incorporates all six legal criteria in the Arizona Constitution to achieve this. Second, the algorithm should be computationally tractable, and we employ a local search method with multilevel coarsening to achieve practical feasibility in computation time. Third, we design the algorithm to be intuitive and familiar to practitioners, drawing on the iterative process used by the AIRC in the 2021 redistricting cycle.

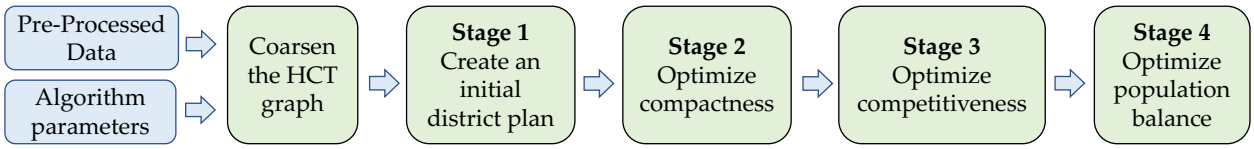


Figure 3.5 An overview of the four stages of the algorithm to draw a district map.

To effectively incorporate Arizona's redistricting criteria in district map design, we decompose the approach into several stages that address each criterion individually. This multi-stage approach is inspired by the redistricting process used by the AIRC, in which an "initial" district map is repeatedly improved while considering public input. Our proposed algorithm consists of four stages, as outlined in Figure 3.5. To alleviate the computational burden in the initial stages, we introduce a coarsening stage that reduces the number of HCTs using the coarsening algorithm described in Section 3.2.5. Section 3.4.1 presents more details on the coarsening phase. Stage 1, described in Section 3.4.2, creates an initial district map that meets the requirements of contiguity, population deviation up to 1%, and the necessary number of majority-Hispanic districts. Stages 2 and 3, discussed in Sections 3.4.3 and 3.4.4, use Recom-based local search to optimize compactness and competitiveness, respectively. The decision to optimize compactness before competitiveness is based on the language of the Arizona Constitution, which stipulates that competitiveness should only be prioritized if doing so does not cause "significant detriment to the other goals" (Arizona, 2021). Stage 4 of the algorithm, described in Section 3.4.5, minimizes the population deviation without compromising the compactness and competitiveness values achieved at the end of stage 3. Section 3.4.6 provides an overall algorithm outline, including its inputs and parameters. An illustration of how a district map evolves through the stages is presented in Section 3.4.7.

The proposed algorithm utilizes different stages that operate at four levels of granularity. Based on empirical investigations, stages 1 and 2 benefit from operating at the coarsest graph size, whereas stages 3 and 4 can find better district maps (concerning competitiveness and population balance, respectively) using the finer and larger graphs. Our approach begins at the hybrid census tract (HCT) level. The 1,440 HCTs are coarsened to a smaller graph size, and stages 1 and 2 operate using this coarsened graph. After stage 2 heuristically optimizes compactness using the coarsened graph, the algorithm uncoarsens the district map back to the HCT level before stage 3 begins. Next, stage 3 improves competitiveness at the HCT level, while permitting a user-specified compromise with compactness. Finally, stage 4 improves population balance after first uncoarsening to the hybrid census block group (HCBG) level and again after uncoarsening to the hybrid census block (HCB) level.

3.4.1 Coarsening Hybrid Census Tracts

Before initiating stage 1, the input size of the census tract graph is reduced using a coarsening method described as follows. This method creates multiple levels of graphs, where each successive level is a smaller version of the preceding level. This is done by merging nodes in one level to create a new node in the next graph level. The multilevel algorithm merges pairs of nodes identified by a graph matching. Given a graph $G = (V, E)$, a *matching* $M \subset E$ is a subset of edges that do not share unit endpoints. Since any matching M can reduce the graph size, we use a matching that prioritizes merging units with low populations, as proposed in Swamy et al., 2023. In particular, we use a *maximal matching* (i.e., one that cannot be locally increased in size). In the special case, a *perfect matching* (i.e., one where every unit is part of a matched edge) reduces the number of coarse level units by half. In reducing the graph size, the goal is to achieve a homogeneous population distribution at the coarse level, as the presence of units with extremely high populations could inhibit the creation of a balanced district map. Such a population-based matching is achieved by assigning a weight to each edge given by the sum of the unit populations and then finding a *minmax weight maximal matching* (MMP). The minmax weight objective minimizes the maximum edge weight within a maximal matching, thereby discouraging the merger of highly populous units. Though MMP is Nondeterministic Polynomial-time NP-Complete, a maximal matching can be heuristically found using the greedy algorithm proposed in Swamy et al., 2023.

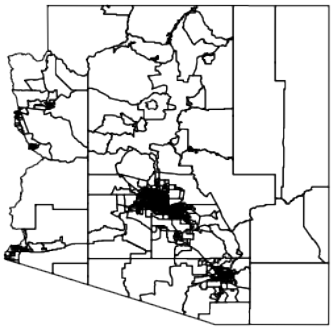
This method uses HCTs as the input to the coarsening procedure (as opposed to HCBGs or HCBs) to respect requirement E of the Arizona Constitution which seeks undivided census tracts to the extent practicable. Figure 3.6 depicts the progression of two levels of coarsening the HCTs. The graph size reduces from $1,440$ HCTs to 805 units in level 1, followed by 493 units in level 2. The level 2 units form the building blocks for drawing district maps in stages 1 and 2.

The number of levels L is a key parameter for the coarsening phase, which controls the scalability of stages 1 and 2. In Section 3.5.1, we illustrate the effect of L on computational time and solution quality (in terms of compactness). Intuitively, larger values of L should lead the algorithm to terminate more quickly. However, we see diminishing results in computational speed using two levels of coarsening instead of one.

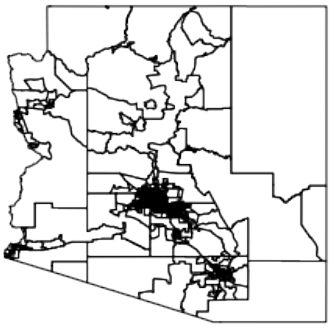
3.4.2 Stage 1: Create an Initial District Map

Stage 1 creates a district map with three basic requirements: contiguity, a specified number of majority-Hispanic districts, and roughly equal population sizes. We set a population deviation tolerance of 1% for district populations to deviate from the ideal population. This 1% deviation serves as a starting point that ensures relatively balanced districts, which will be further refined in stage 4 (up to a deviation as small as one person). We further divide stage 1 into two phases: first create contiguous and 1%-balanced districts and then create the requisite number of majority-minority districts.

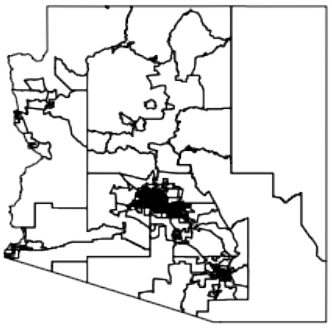
In the first phase, a contiguous and 1%-balanced district map is drawn using a local search method proposed by Bozkaya et al., 2003. This is a *multi-kernel growth* method, where the units are first randomly grouped into contiguous districts without requiring the districts to be equipopulous. Then, the population balance is improved using Flip-based local search. Each iteration in the local search checks whether a randomly chosen unit can be transferred from its current district to a neighboring district. To maintain the contiguity of the current district, we verify whether the subgraph induced by the district remains connected after the potential removal of the unit, using depth-first search. If the transfer maintains contiguity and the population balance of the map improves, we update the map to be the current best map. This procedure



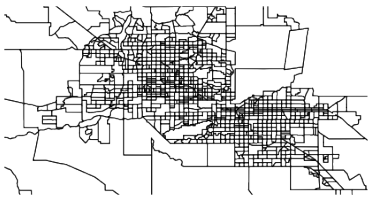
(a) Arizona's dual hybrid census tract graph with 1,440 units.



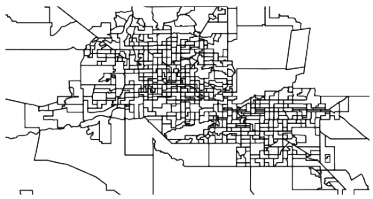
(b) Arizona after one level of coarsening with 805 units.



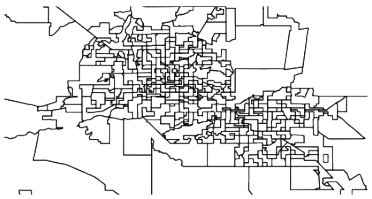
(c) Arizona after two levels of coarsening with 493 units.



(d) Hybrid census tracts in the city of Phoenix.



(e) Phoenix after one level of coarsening.



(f) Phoenix after two levels of coarsening.

Figure 3.6: Two levels of coarsening Arizona's hybrid census tract graph; a reduction of size from 1,440 units to 493 units.

continues until the population balance cannot be further improved using Flip-based local search. At the end of this procedure, if the district populations of the district map are within the 1% deviation tolerance, then this contiguous and 1%-balanced district map moves on to the next phase; otherwise, the multi-kernel growth is repeated from scratch until the populations are within the 1% deviation tolerance.

In the second phase, we propose an iterative method to create the requisite number of VRA districts, N_{VRA}, each with a minority fractional threshold α_{VRA}. Let $Q \subseteq \{1, 2, \ldots, K\}$ be the subset of VRA districts in the current district map. If $|Q| \geq N_{\text{VRA}}$, this phase is skipped. Otherwise, the algorithm seeks to build one new VRA district iteratively until $|Q| \geq N_{\text{VRA}}$. Each iteration uses Recom-based local search to maximize a *VRA score* defined as the fractional minority population in the non-VRA district with the most minority population. For a given district map z, this method maximizes

$$\text{VRA score } \phi_{\text{VRA}}(z) = \max_{k=1,2,\ldots,K:k\neq Q} p_k^{\min}/P_k, \tag{3.4}$$

where $p_k^{\min}$ and P_k are the minority and total populations in district k, respectively. Every Recom iteration randomly selects two neighboring districts and redraws their district boundary by bisecting a randomly generated spanning tree in the merged subgraph of the two districts. Maximizing the VRA score would improve the district map in a direction where a district is closer to becoming a VRA district. When the VRA score crosses α_{VRA}, the algorithm has found one new VRA district, and Q is updated to include this district. Otherwise, if the algorithm proceeds for a certain number of iterations without any improvement in the VRA score, this stage is restarted from scratch. Finally, this phase terminates when $|Q| \geq N_{\text{VRA}}$.

The pseudocode for this stage is presented in Appendix B.1.2. After both phases, stage 1 returns a district

map that is contiguous, 1%-balanced in population and contains the requisite number of majority-minority districts. However, the shapes of the districts after stage 1 may be non-compact, and hence must be improve in stage 2.

3.4.3 Stage 2: Optimize Compactness

The goal of stage 2 is to enhance the compactness of the district map created in stage 1 while also transforming the district map from the coarsened level to the HCT level. Compactness is measured by the sum of shared district boundaries as in Equation (3.2). Optimizing compactness in stage 2 uses the Recom-based local search method. This maximum number of consecutive iterations without improvement is a key parameter in stage 2, which determines how long we allow the local search to seek more compact district maps, trading off between improved compactness and a faster computational time. Furthermore, note that each iteration preserves contiguity, 1%-balance, and the requisite number of VRA districts. The pseudocode for locally optimizing compactness is presented in Appendix B.1.1.

After heuristically optimizing compactness in the coarse level, the district map is iteratively uncoarsened to the HCT graph level. Here, the units in the finer level are assigned to the same district as their corresponding coarsened units. Each level of uncoarsening further improves compactness using Recom with the same termination criterion as the coarsest level. After repeating this process for L levels of uncoarsening and compactness optimization, stage 2 returns a locally optimized compact district map.

3.4.4 Stage 3: Optimize Competitiveness

Stage 3 shifts the focus from compactness to competitiveness. As specified in the Arizona Constitution, competitiveness is treated as a secondary objective prioritized as long as it does not cause a significant deterioration in the other goals. To improve competitiveness, the algorithm makes a calculated *compromise* on the compactness achieved in stage 2, using a user-specified parameter. This allows the algorithm to strike a balance between achieving a competitive district map and maintaining the compactness of the map within a specified limit.

Stage 3 maximizes the competitiveness score defined in Equation (3.3), which measures the number of districts with a vote spread less than 7%, while minimizing the maximum vote spread among all districts. This stage uses a Recom-based local search method, while also considering the constraints of contiguity, 1%-population deviation, and the required number of majority-Hispanic districts. Additionally, it imposes a fourth constraint that the compactness score from Equation (3.2) should not be worse than a user-specified fraction above the heuristically optimized compactness value found at the end of stage 2. Specifically, the constraint requires that the compactness of the resulting map is less than or equal to $COMP \times (1+\delta)$, where $COMP$ is the compactness value from stage 2 and $\delta \geq 0$ is the *compromise factor*, which determines how much the algorithm is permitted to compromise compactness in the pursuit of competitiveness. Larger values of δ prioritize competitiveness, while smaller values prioritize compactness.

The compromise factor is a key parameter in stage 3. Section 3.5.2 provides an empirical analysis for different values of the compromise factor ranging from 0% to 30%, and measures its effect on the trade-offs between compactness and competitiveness. As in stage 2, another key parameter here is the termination criterion for the local search, determined by the maximum number of consecutive iterations without improvement. The pseudocode for this stage is presented in Appendix B.1.1. The district map produced after stage 3 is locally optimized for competitiveness subject to the δ-compromise constraint in

compactness.

3.4.5 Stage 4: Optimize Population Balance

The final stage of the algorithm minimizes the imbalance in district populations in the stage 3 map, which can deviate by up to 1% of the ideal population size or up to 7,946 people in Arizona. While doing so, stage 4 does not compromise the compactness and competitiveness scores achieved by the district map produced at the end of stage 3. To enable greater flexibility in improving population balance, the stage 3 district map is transformed from the HCT level to more fine-grained levels, first to the hybrid census block group (HCBG) level, and then to the hybrid census block (HCB) level. Arizona has 3,880 HCBGs and 110,778 HCBs, preserving the regions designated in Section 3.3.5. Using these finer geographical units allows the algorithm to make adjustments at a more detailed level and bring the population deviation as close to zero as possible.

Starting from the map found in stage 3, this map is transformed to the HCBG level and then a Flip-based local search method (with the pseudocode presented in Appendix B.1.1) is used to locally optimize population balance, which moves one randomly chosen unit in each iteration to a neighboring district. The objective is to minimize the maximum deviation in district populations from the ideal district population. Flip-based local search is preferable to Recom, since the large number of geographical units in the HCBG and HCB levels makes it computationally intractable to use Recom for each iteration. After optimizing population deviation in the HCBG level, the map is transformed to the HCB level, and Flip-based local search is repeated in the HCB level.

As with the previous stages, the key parameter in stage 4 is the termination criterion given by the maximum number of consecutive local search iterations with no improvement. Setting this parameter to be 1,000 iterations, the resulting district maps reported in Section 3.5.3 have deviations of at most three people from the ideal district population, making them legally viable according to the six criteria specified in the Arizona Constitution.

3.4.6 Outline of the Optimization Framework

Algorithm 3.1: Outline of the Optimization Framework

Inputs : Hybrid census tract graph G_{HCT}, hybrid census block group graph G_{HCBG}, hybrid census block graph G_{HCB}, number of districts $K \in \mathbb{Z}^+$, requisite number of majority-minority districts $N_{\text{VRA}} \in \{1, 2, \ldots, K\}$, minority fractional threshold $\alpha_{\text{VRA}} \in [0, 1]$

Parameters: Number of coarsening levels $L \geq 0$, maximum number of consecutive local search iterations with no improvement $N^s_{\text{iter}} \in \mathbb{Z}^+$ for stage $s = 1, 2, 3, 4$, compromise factor $\delta \geq 0$

Output : District map z^*

$G_{\text{coarse}} \leftarrow$ Coarsen G_{HCT} by L levels

$z_{\text{initial}} \leftarrow$ Stage 1: Create an initial district map in G_{coarse} with K districts (with N_{VRA} VRA districts, each with α_{VRA} minority population) that are connected and 1%-balanced

$z_{\text{compact}} \leftarrow$ Stage 2: Optimize compactness of z_{initial} in G_{coarse} and uncoarsen to G_{HCT}

$z_{\text{cmpttv}} \leftarrow$ Stage 3: Optimize competitiveness of z_{compact} in G_{HCT} considering compromise factor δ

$z^* \leftarrow$ Stage 4: Uncoarsen z_{cmpttv} first to G_{HCBG} and then to G_{HCB}, while optimizing population balance in both graphs

return z^*

Algorithm 3.1 outlines the stages of the algorithm. We now describe how the algorithm parameters are used in each stage. Line 1 summarizes the coarsening stage, which coarsens G_{HCT} for L levels. Lines 2 to 5 summarize stages 1 to 4. As described in Sections 3.4.2-3.4.5, stages 1 and 4 use Flip-based local search to locally optimize population balance. Further, stages 1, 2 and 3 use Recom-based local search to locally optimize the VRA, compactness and competitiveness scores, respectively. For each stage $s = 1, 2, 3, 4$, the maximum number of consecutive local search iterations without improvement N^s_{iter} controls the trade-off between solution time and the objective value achieved. Note that for stage 1, N^1_{iter} applies only to its second phase and not the first phase. Furthermore, for stage 3, the compromise factor δ controls the trade-off between optimizing compactness over competitiveness. Section 3.5 analyzes the impact of L and δ on compactness and competitiveness, and the solution time. The parameters in stages 1-3 can be adjusted using the web application located at `https://redistricting.cs.illinois.edu/#optimap`. Note that the computational time to run the algorithm within the web application varies based on the web host's bandwidth, potentially resulting in longer computational times compared to those reported in Section 3.5.

3.4.7 An Illustrative Example

Figure 3.7 illustrates a district map's evolution through the algorithm's four stages. For each map, we show the district boundaries for the nine districts, and three metrics for each district — district compactness, vote spread, and the percent deviation from the ideal population. Here, district compactness is measured by the total boundary of each district shared with its neighboring districts. Note that the compactness score optimized in stage 2 is half the sum of the district compactness (since the boundaries are counted twice by every pair of neighboring districts). The algorithm parameters are set as follows: $L = 1$, $\delta = 0.2$, and $N^s_{\text{iter}} = 500$ for $s = 1, 2, 3$ and $N^4_{\text{iter}} = 1,000$. All the computations in this chapter are conducted on an Apple M1 machine with 16 GB memory and the code is written in Python 3.9.12.

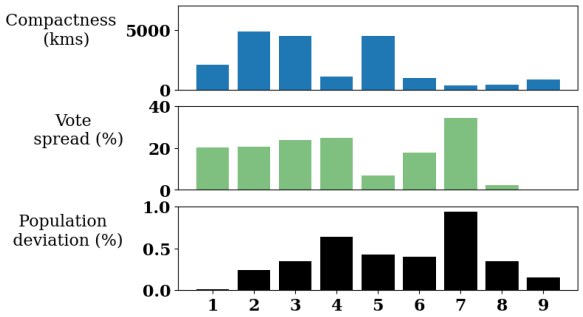
Figures 3.7a, 3.7c, 3.7e and 3.7g show the district maps after stages 1, 2, 3 and 4, respectively. After optimizing compactness in stage 2, the compactness score has improved from 9,735 km to 4,309 km (compare Figures 3.7b and 3.7d). However, the perimeter lengths of two districts (labeled 4 and 8) have slightly increased. This improvement in overall compactness is also reflected visually in the map in Figure 3.7c. After optimizing competitiveness in stage 3, the vote spreads have reduced (compare Figures 3.7d and 3.7f), while the compactness score is allowed to worsen by 20%. The number of competitive districts has increased from *four* after stage 2 to *seven* after stage 3. After optimizing population deviation in stage 4, the maximum deviation in district population is reduced from 7,471 people to *five* people (compare Figures 3.7f and 3.7h).

Executing stages 1-3 is relatively quick, whereas stage 4 requires more computational time. Stages 1-3 collectively took four minutes, whereas stage 4 took 47 minutes, owing to stage 4's computational burden of executing local search in the large HCB level.

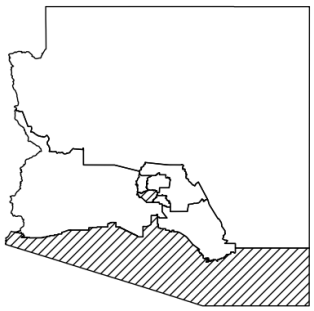
3.5 Results and Curated Maps

The proposed algorithm incorporates several random elements, so it is important to study how the selected values for the algorithm parameters influence the algorithm's average performance as measured by the solution quality (i.e., compactness and competitiveness) and the run time. This study focuses on two key parameters: the number of coarsening levels in stages 1 and 2, and the compromise factor δ in stage 3. The maximum number of consecutive iterations without improvement before the local search terminates is also a key parameter whose value is fixed in this study. Section 3.5.1 presents a preliminary investigation on how the

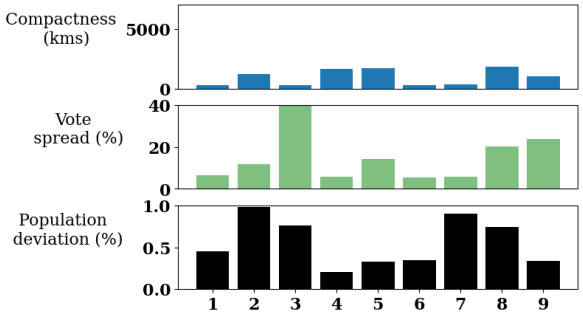
(a) Stage 1: Initial map.



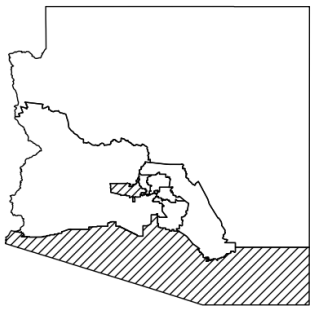
(b) Metrics in the nine districts after stage 1.



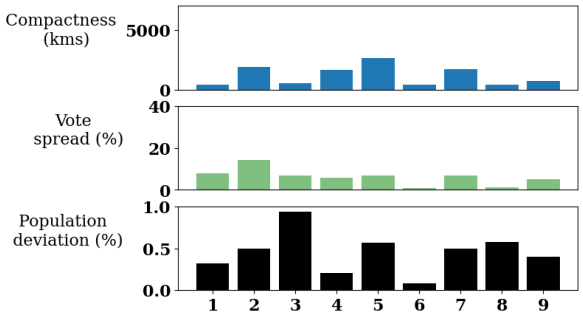
(c) Stage 2: Compact map.



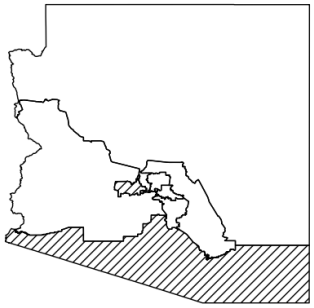
(d) Metrics in the nine districts after stage 2; compactness is improved.



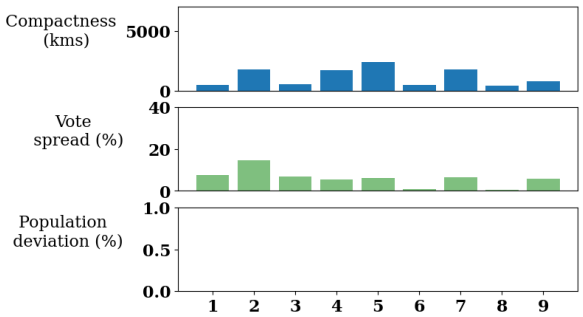
(e) Stage 3: Compact and competitive map.



(f) Metrics in the nine districts after stage 3; vote spreads are improved.



(g) Stage 4: Compact, competitive and balanced map.



(h) Metrics in the nine districts after stage 4; population deviations are improved.

Figure 3.7: The evolution of a district map through the four stages of the algorithm; after stage 4, the district map is contiguous, balanced in population, compact, and competitive, while respecting the preserved regions and including two majority-Hispanic districts (shaded).

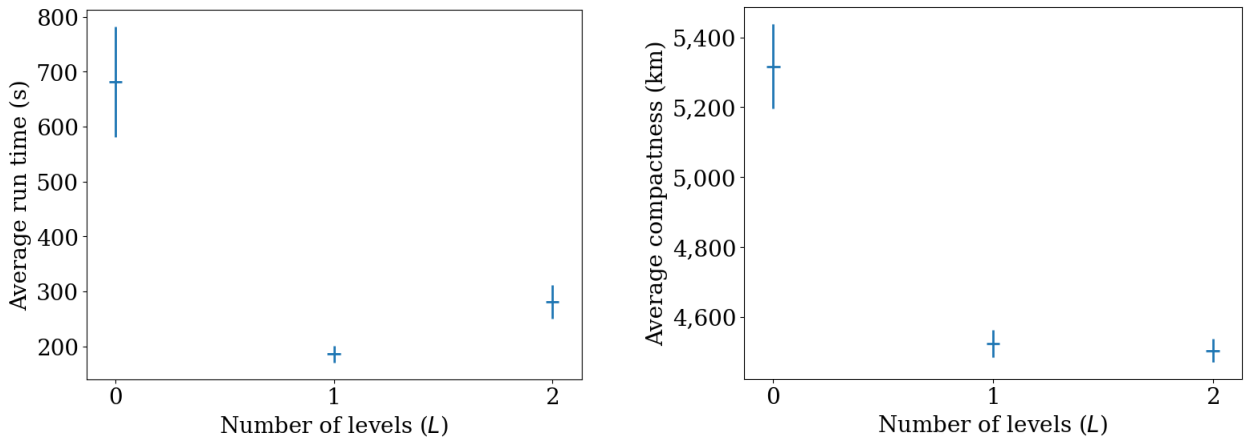
number of coarsening levels (L) affects the run time and the compactness of the generated district maps in stages 1 and 2. We set $N^s_{\text{iter}} = 100$ for $s = 1, 2$, which is sufficient for this preliminary analysis. Next, Section 3.5.2 explores how the compromise factor δ affects the trade-off between compactness and competitiveness objectives in stage 3. For each value of $\delta = 0, 0.1, 0.2, 0.3$, we run stages 1-3 of the algorithm 100 times to obtain 100 maps at the end of stage 3. To accommodate better objective values in this investigation, N^s_{iter} is expanded to 500 iterations for $s = 1, 2, 3$.

To further demonstrate the effectiveness of the proposed optimization algorithm, we present a curated set of three maps with varying trade-offs on compactness and competitiveness. Section 3.5.3 summarizes these maps after stage 4 is applied to locally optimize the population balance. Here, N^4_{iter} is set to $1,000$ to achieve a population deviation as small as one person as possible. Section 3.5.4 discusses these maps in more detail, and Section 3.5.5 highlights the key insights gained from applying the algorithm in Arizona.

3.5.1 Stages 1-2: Picking the Appropriate Number of Coarsening Levels

Recall that stages 1 and 2 draw the most compact map while retaining feasibility with respect to three hard constraints: contiguity, 1%-population balance, and creating two majority-Hispanic districts. The coarsening phase is the pre-processing stage before stage 1 that reduces the size of a large input graph into a smaller size. The number of coarsening levels, denoted by L, decides the extent of coarsening. We now investigate the impact of coarsening on the algorithm's performance in terms of the compactness score and the run time.

We investigate three settings — from zero to two levels of coarsening, i.e., $L = 0, 1$, and 2. The goal is to assess which setting produces district maps with the fastest run time and the best compactness score. To achieve this goal, for each value of L, we run stages 1 and 2 of the algorithm for 100 independent runs and collect the 100 district maps produced at the end of stage 2. Within each run, we set $N^1_{\text{iter}} = N^2_{\text{iter}} = 100$.



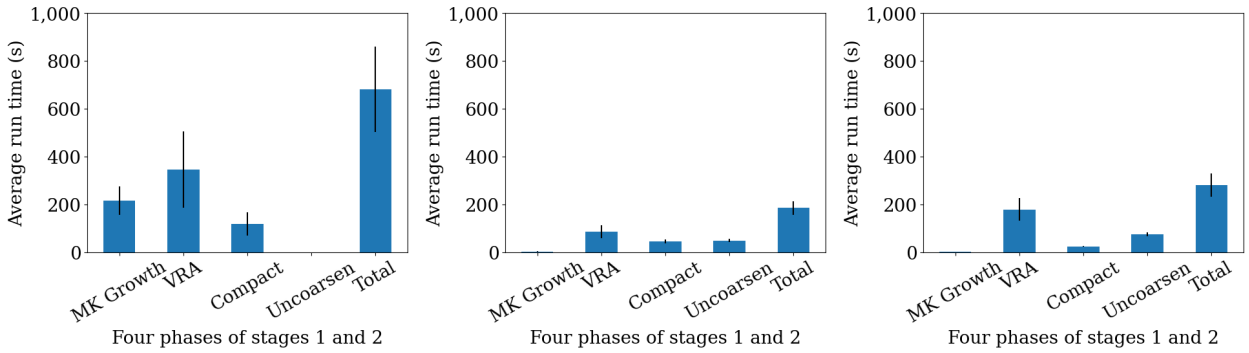
(a) Average run time vs. number of coarsening levels.

(b) Average compactness vs. number of coarsening levels.

Figure 3.8: Average run times and the compactness scores along with their 95% confidence intervals based on 100 independent runs of the algorithm for three choices of the number of coarsening levels.

We are interested in analyzing how L affects the run time and the compactness score. Figure 3.8 depicts 95% confidence intervals for the average run time and the average compactness. On average, one level of coarsening runs faster and produces more compact district maps when compared to no coarsening. The average compactness score is $5,317$ km with no coarsening, whereas it is $4,523$ km with one level of coarsening,

producing a 15% reduction. We observe that the average run time with no coarsening (681s) is more than three times the average run time with one level of coarsening (186s). Additionally, we note that two levels of coarsening do not produce any significant advantage in either compactness or run time compared to one level of coarsening.



(a) Without any coarsening. (b) With one level of coarsening. (c) With two levels of coarsening.

Figure 3.9: Average run times and their 95% confidence intervals for the five phases of stages 1 and 2 of the algorithm aggregated for 100 independent runs for three choices of the number of coarsening levels (L).

To understand the differences in run times, we report a breakdown of the time taken in the various sub-parts of the algorithm. Figure 3.9 depicts the run times for the algorithm's four phases during stages 1 and 2. Recall that stage 1 of the algorithm has two phases – first, it finds a contiguous and 1%-balanced district map using multi-kernel growth, and then creates the requisite number of VRA districts. In Figure 3.9, we denote the first phase as "MK Growth" and the second as "VRA." Furthermore, stage 2 first optimizes compactness using local search in the coarsest level, and then uncoarsens the map to finer levels while applying local search in each level. We denote the coarsest-level local search as "Compact" and the local search during uncoarsening as "Uncoarsen."

Notice that without coarsening, stage 1 of the algorithm, which includes MK Growth and VRA, takes longer to execute on average compared to scenarios that apply coarsening. This is due to the difficulty of finding a compact, 1% balanced district map with two VRA districts in the large HCT level graph. The use of coarsening demonstrates a scalability advantage. Additionally, we can see worsening returns with two levels of coarsening. In particular, VRA takes longer to execute with two levels of coarsening ($L = 2$) than with one level ($L = 1$) because the coarseness of the level 2 graph makes it more difficult to create the necessary number of VRA districts, leading to more iterations. Based on these findings, we conclude that for Arizona's HCT graph, one level of coarsening provides the best computational advantage to the algorithm compared to no coarsening and two levels of coarsening. In the subsequent sections, we will use $L = 1$ in stages 1 and 2 of the algorithm.

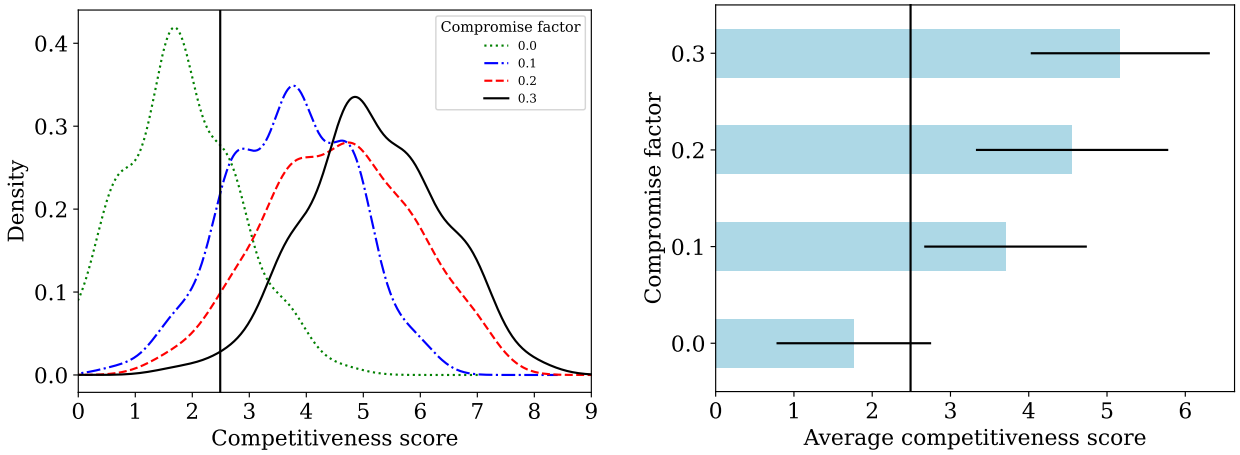
3.5.2 Stages 1-3: Compromise Between Compactness and Competitiveness

Recall that stage 3 of the algorithm optimizes competitiveness, while ensuring that the compactness score does not worsen by more than a compromise factor (δ) times the compactness score found at the end of stage 2. To empirically investigate the trade-off between competitiveness and compactness, we explore four different values for $\delta \in \{0, .10, .20, .30\}$ and observe their effect on the two objectives. We run stages 1-3 with each stage terminating at 500 iterations of no improvement, allowing the algorithm to explore more solutions

and potentially achieve better compactness values compared to the analysis in the previous section. With these settings, we perform 100 runs of stages 1-3 of the algorithm and collect the 100 resulting district maps. This section offers insights into these district maps and compares their objective values to those achieved by CD118.

It is important to note that the 100 maps presented here have not been optimized for population balance yet (i.e., stage 4 has not been performed), so the comparisons with CD118 focus solely on compactness and competitiveness. For a more direct comparison with CD118, applying stage 4 would be ideal. However, due to the significant computational time required to execute stage 4 for all 100 maps, we have omitted its application. Nonetheless, since stage 4 does not allow a map's compactness score or competitiveness score to worsen while it improves population balance, the scores that these 100 maps achieve would not be worse than the scores analyzed here, though the degree of population balance that could be achieved for each map is not yet clear. To examine the ability of the algorithm to improve population balance while maintaining the compactness and competitiveness scores found at the end of stage 3, we apply stage 4 to three curated maps and compare them with CD118 using various metrics in Section 3.5.3.

On average, the competitiveness score improves as the value of δ is increased, as expected. Figure 3.10a depicts the distribution of competitiveness scores across the 100 district maps obtained for each of the four values of δ. Figure 3.10b shows how the average competitiveness improves with increasing values of δ. Note that when $\delta \geq 0.1$, the average competitiveness value is better than that of CD118.



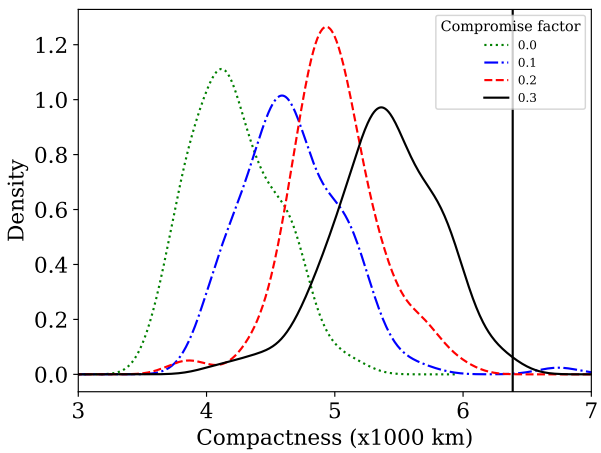
(a) Kernel density estimate (KDE) plot of **competitiveness** for 100 maps at each compromise factor.

(b) Average **competitiveness** (along with the 95% confidence intervals) over 100 maps at each compromise factor.

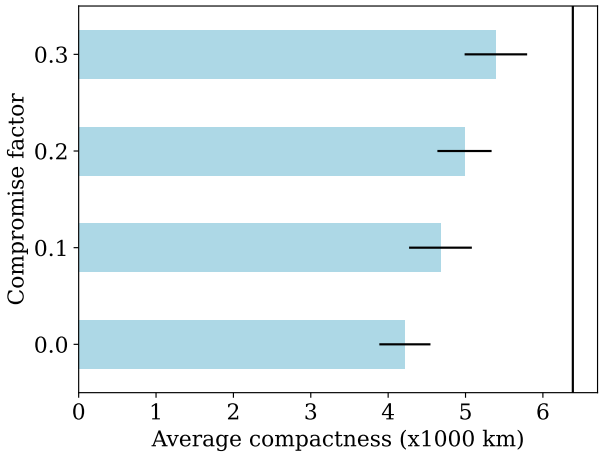
Figure 3.10: The effect of the compromise factor on the competitiveness of the maps after stages 1-3. The vertical solid line represents the competitiveness score of the enacted map (CD118).

Also as expected, larger values of δ produce district maps with worse compactness values on average. Figure 3.11a depicts the distribution of compactness values across the 100 district maps obtained for each compromise factor. Figure 3.11b shows how the average compactness worsens for increasing compromise factor values. Note that for all values of δ considered, 397 out of the 400 maps have better compactness than that of CD118.

The majority of the 400 maps produced *dominate* CD118; a map dominates another map if it has better values for both compactness and competitiveness. Figure 3.12a plots the two objective values for the 400 maps; 80% of the maps dominate CD118, and CD118 dominates none. Among maps produced with a compromise

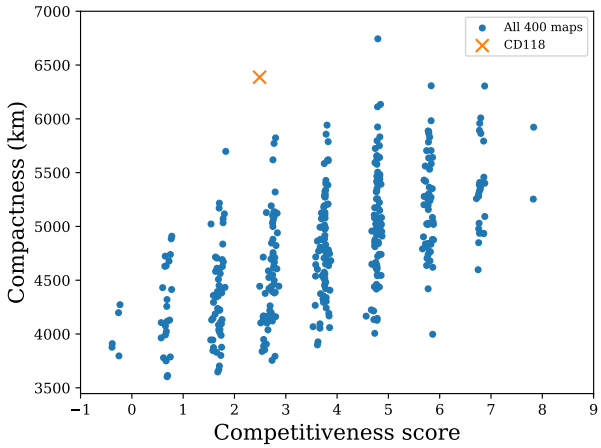


(a) KDE plot of **compactness** for 100 maps at each compromise factor.

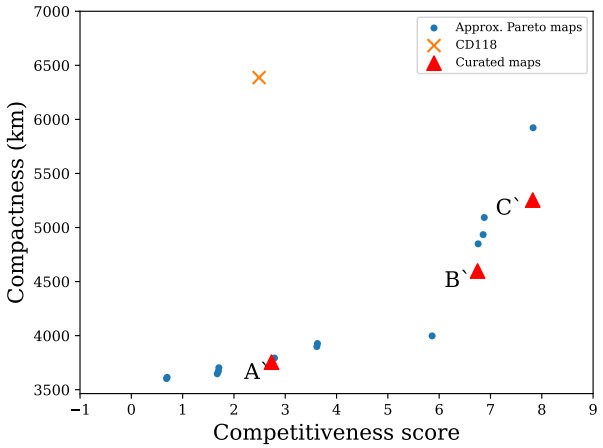


(b) Average **compactness** (along with the 95% confidence intervals) over 100 maps at each compromise factor.

Figure 3.11: The effect of the compromise factor on the compactness of the maps after stages 1-3. The vertical solid line represents the compactness of the enacted map (CD118).



(a) Competitiveness vs. compactness for all 400 maps.



(b) Competitiveness vs. compactness for 16 non-dominated district maps.

Figure 3.12: Trade-off between competitiveness and compactness based on the 400 district maps generated by the algorithm after stages 1-3. The maps are contrasted against the enacted district map (CD118).

factor greater than 0.1, 96% of the maps dominate CD118. Furthermore, Figure 3.12b plots a subset of 16 *non-dominated* district maps; a map is non-dominated if no other map (among the 400) dominates it.

3.5.3 Three Curated Maps

Among the 16 non-dominated maps from Figure 3.12b, we now focus on three maps labeled A', B' and C', which have the smallest compactness with three, seven and eight competitive districts, respectively. Map A' has the same number of competitive districts as CD118, while B' and C' represent the maximum levels of competitiveness achieved. The three maps emphasize different degrees of trade-off between compactness and

competitiveness. To allow a more direct comparison with CD118, Stage 4 of the algorithm is then applied to improve the population deviation of each of these maps beyond the 1%-balance level. This stage first runs at the HCBG level and then at the HCB level, without compromising the compactness and competitiveness obtained at the end of stage 3. The termination criteria for both levels is set to $N_{\text{iter}}^4 = 1,000$ iterations of no improvement. Let A, B and C denote the respective maps after stage 4 has been applied to maps A', B' and C'.

Table 3.1: An overview of three curated district maps. *A district is competitive (vs. safe) if the difference between the Democratic and Republican voters is less than (vs. greater than or equal to) 7% of the total voters in the district. **The 118th Congressional district map (2023-2025) was drawn in 2022.

District Map	Compactness (km)	Number of competitive* districts	Maximum Vote Spread	Number of safe* Dem. vs. Rep. districts	Maximum population deviation from ideal
A (most compact)	3,755	3	26.77%	3D,3R	1
B	4,597	7	25.28%	1D,1R	3
C (most competitive)	5,253	8	18.07%	0D,1R	3
118th Congressional map**	6,383	3	51.69%	2D,4R	1

Maps A, B and C are the final district maps that cater to all six of Arizona's redistricting requirements to a practicable extent. Table 3.1 gives a summary of metrics for the three district maps as well as CD118. All three maps have better compactness scores than CD118, while maintaining at least as many competitive districts, and a smaller maximum vote spread. Among the non-competitive (or "safe") districts, the three maps also yield a proportional share of districts to the two parties.

3.5.4 Discussion

In this section, we discuss each map presented in Section 3.5.3. Map A emphasizes compactness, whereas maps B and C emphasize competitiveness. Appendices B.2, B.3 and B.4 present figures and demographic information about the three maps; district labels refer to those presented in the appendices when discussing each map.

Map A is the most compact map, with three competitive districts. It has a compactness score of 3,755 km and a maximum vote spread of 26.77%. It has six safe districts, among which three districts have Democratic majorities and three have Republican majorities. Six districts (labeled 1, 3, 5, 6, 7, 9) share parts of Phoenix. Of these, two districts (labeled 1 and 7 with respective vote spreads 17.15% and 18.33%) pack the Republican voters in the region surrounding Phoenix, while two other districts (labeled 5 and 9 with respective vote spreads 26.47% and 26.77%) pack the Democratic voters in the city of Phoenix. There are three competitive districts (labeled 2, 3 and 6 with respective vote spreads 5.8%, 5.5% and 3.8%). Two districts (labeled 1 and 4) share parts of Tucson, where one district (labeled 1 with a vote spread of 17.15%) packs the Republican voters, while the other district (labeled 4 with a vote spread of 13.15%) packs the Democratic voters.

Map B is less compact than map A, but it has four more competitive districts. Seven of the nine districts are competitive, while the other two have a maximum vote spread of 25.28%. The compactness score is 4,597 km. All the districts include parts of Phoenix or Tucson. Among the two safe districts, one district (labeled 2 with a vote spread of 25.28%) has a Republican majority and the other district (labeled 8 with a vote spread of 13.28%) has a Democratic majority. Six districts (labeled 1, 2, 3, 5, 8, 9) share parts of Phoenix,

while three districts (labeled 4, 6, 7) share parts of Tucson. Further, five districts (labeled 1, 3, 5, 8, 9) are contained entirely inside Maricopa County.

Map C is the most competitive, but the least compact map among the three maps presented. It has eight competitive districts, and the one safe district has a vote spread of 18.07%. This high level of competitiveness is achieved by evenly dividing the Democratic voters in Phoenix and Tucson with the Republican voters dispersed in the surrounding regions. In particular, Phoenix is divided among six different districts (labeled 3, 4, 6, 7, 8, 9), and Tucson is divided among the three remaining districts (labeled 1, 2, 5). Moreover, five districts sharing the Phoenix area (labeled 3, 4, 6, 8, 9) are contained entirely within Maricopa County, effectively dividing the Democratic voters in urban Phoenix and the Republican voters in suburban Phoenix. Among the three maps presented in this report, map C is also the least compact with a compactness score of 5,253 km.

3.5.5 Key Insights

This report's three congressional district maps illustrate the trade-offs between competitiveness and compactness. In this section, we discuss some key insights that can be drawn from these maps.

The most compact map A creates six safe districts, with three for each party. In contrast, maps B and C, which are designed to be highly competitive, balance Democratic voters in urban centers and Republican voters outside of them to maintain competition in a larger number of districts. Achieving this degree of balance results in the creation of non-compact districts. Thus, examining these three maps shows the trade-off between compactness and competitiveness. Note that the compactness objective has resulted in the three maps preserving an additional county, Yuma, which was not preserved in CD118.

The Arizona Constitution does not place any emphasis on partisan fairness beyond competitiveness. However, to assess the partisan consequences of adopting these maps, we analyze the maps' scores with respect to the well-known measures of partisan fairness discussed in Section 3.2.1, namely the efficiency gap (EG) and partisan asymmetry (PA). See Chapter 2 for their mathematical definitions. Recall that EG measures the difference in wasted votes between the two parties, while PA is related to the difference in the number of seats gained or lost by the two parties when there are fluctuations in their share of voters. Table 3.2 presents the partisan fairness values for these district maps, and for CD118 for contrast.

Table 3.2: Partisan fairness metrics for the presented maps. *The 118th Congressional district map (2023-2025) was drawn in 2022.

District Map	Efficiency Gap	Partisan Asymmetry	Number of Dem. vs. Rep. districts
A (most compact)	19.73%	4.12%	3D,6R
B	3.86%	2.71%	5D,4R
C (most competitive)	9.37%	2.20%	4D,5R
118th Congressional map*	22.01%	7.11%	3D,6R

A smaller EG value indicates more proportionally divided district majorities. All three curated maps have smaller EG values compared to CD118. Specifically, map B has the smallest EG value, which yields a split of 5-4 in districts between the Democratic and Republican parties, respectively. The high EG value in compact map A and in CD118 can be attributed to the packing of urban Democratic voters in a few districts, which leads Democrats to waste significantly more voters than Republicans. This degree of packing suggests that

optimizing compactness, as was the goal for map A, is detrimental to ensuring equity in partisan outcomes. See Swamy et al., 2023 and Dobbs et al., 2023 for optimization methods that explicitly optimize EG and PA.

Furthermore, CD118 has a relatively high value of PA compared to all three maps. This suggests that with fluctuations in voters, CD118 would yield an asymmetric advantage to the Republican party compared to the Democratic party. However, as the maps become more competitive (from A to C), the value of PA becomes smaller.

3.6 Generalizability and Limitations

Despite being tailored specifically to Arizona's constitutional redistricting requirements, the algorithmic framework presented in this study can capture many of the redistricting requirements of other states. The criteria considered in this study, such as contiguity, compactness, preservation of communities of interest, preservation of political boundaries, and competitiveness, are commonly imposed in other states. For example, among the 43 states that redrew their congressional districts following the 2020 Census (i.e., those apportioned more than one district), contiguity is required in 23 states, compactness is required in 18 states, preservation of communities of interest is imposed in 13 states, preservation of political boundaries is required in 19 states, and competitiveness is emphasized in five states (Ballotpedia, 2021b). Additionally, population balance and the creation of majority-minority districts are federal requirements that apply to all 43 states. There are other criteria required in some states that are not considered in this framework, although the framework is likely flexible enough to accommodate them. For example, seven states require that districts "preserve cores of prior districts." This requirement can be handled with a pre-processing stage that preserves pre-defined regions in prior districts, similar to preserving communities of interest. Therefore, in terms of criteria, the algorithmic framework presented in this study can be adapted to meet the redistricting requirements of these states.

There are several limitations to consider when applying the framework presented in this chapter in a practical setting. One notable limitation is that our framework relies on complete transparency in the process, criteria and data used. This is possible when a state uses an independent commission to draw district maps, such as in Arizona. However, only a minority of states, nine out of 43, employed non-partisan redistricting commissions following the 2020 Census (Ballotpedia, 2020b). The majority of states instead had their districts drawn by the majority party in their state legislature, making the adoption of a non-partisan framework such as ours less likely in these states. A second notable limitation is that while our framework prioritizes population balance to a practicable extent, the resulting district maps have deviations of up to three people from the ideal district population. While this deviation may be considered high by a proponent of a stricter deviation (of one person), note that by loosening the constraints on compactness and competitiveness in stage 4, it may be possible for the proposed algorithm to achieve better population balance. A third limitation is that our framework, being a heuristic method, does not explore the entire solution space to find an optimally compact or competitive map. The specific trajectory traversed by the heuristic might overlook better district maps. Moreover, the coarsening phase induces an approximation of the original graph. For a broader exploration of the solution space, we recommend increasing the number of local search steps and/or limiting the number of coarsening levels beyond the settings used in this chapter. The fourth limitation pertains to the use of CD118 as a baseline, which may limit the potential for achieving better criteria, particularly concerning preserved communities of interest and political subdivisions.

3.7 Conclusions and Future Research Directions

This chapter presents a case study in applying an optimization framework for creating legally valid district maps in Arizona, which addresses multiple criteria and handles large input sizes by traversing multiple granularities of input graphs. The district maps presented in this study demonstrate the effectiveness of the optimization algorithm for political redistricting in practice. Arizona's enacted map CD118 serves as a benchmark for comparison with the maps generated by the algorithm, revealing that better compactness and competitiveness can be achieved using the optimization framework.

The trade-off between different criteria must be carefully considered when designing an optimization algorithmic approach to optimize specific criteria. Stage 3 of the proposed algorithm includes a user-specified parameter to capture the trade-off between compactness and competitiveness. The trade-offs between other criteria can also be explored. For example, preserving certain counties, as in the case of CD118, may impede the optimization of other criteria, such as compactness.

This framework provides a foundation for further research on practical optimization frameworks for redistricting. Potential directions for future research include (i) investigating other ways of arranging the stages (such as optimizing competitiveness before compactness), (ii) incorporating additional considerations, such as limiting the extent of modifications as a map progresses through the stages or adding objectives such as minimizing county splits, (iii) adapting the framework to other states by incorporating additional criteria, and (iv) evaluating variants of local search such as simulated annealing or hill climbing. Exploring these research directions could enable optimization algorithms to become practical and effective tools in future redistricting processes.

Chapter 4

Heuristics For Game-Theoretical Districting Protocols

4.1 Introduction

Political districting involves multiple stakeholders, including political parties, independent commissions, candidates, and voters. In Chapters 2 and 3, optimization frameworks are presented wherein a central authority, such as an independent commission, formulates a set of objectives and constraints, and subsequently optimizes them. An alternative perspective involves framing districting as a game among political parties, referred to as "players," where each player's overall objective is to maximize the number of districts wherein they have a simple majority of voters. This approach aims to devise a game protocol that allows players to pursue self-interested strategies in district creation while ensuring that the resulting district plan, under equilibrium play, performs well according to fairness metrics.

Several game protocols have been introduced in the literature, typically involving two political parties. Landau and Su, 2014 propose a fair division protocol where an independent arbiter first splits a state into two pieces, and allows each party to draw districts in each piece. Pegden et al., 2017 introduce the *I-cut-you-freeze* (ICYF) protocol, where players take turns creating districts from the remaining region (starting with the entire state) while also freezing one of these districts alternately. Ludden et al., 2023 propose a *bisection* protocol where players alternatively bisect the remaining region (starting from the whole state) further into two smaller pieces. Notably, the ICYF and bisection protocols offer an advantage over the fair division protocol proposed by Landau and Su, 2014 as they eliminate the need for an arbiter's involvement.

These studies have derived closed-form equilibrium strategies for a simplistic game setting, called the *continuous non-geometric* (CN) setting. In this setting, the game relies solely on two inputs: (i) the overall vote-share of each party, represented as a number between zero and one, and (ii) the number of districts denoted as $K \geq 2$. The task is to allocate this vote-share among the K districts. For the CN setting, these works quantified the fairness implications of the resulting district map when the players execute their equilibrium strategy. In particular, the ICYF in some case yields a substantial advantage to one party. On the other hand, the bisection protocol yields more proportional districts than ICYF, even though ICYF yields smaller efficiency gap (EG) and partisan asymmetry (PA) values.

Unlike the CN setting, practical districting, as modeled in Chapter 2, involves discrete inputs due to the availability of population and voter data within discrete and indivisible geographical units, such as census

tracts. Furthermore, districts must be geographically contiguous. This setting of the game is called the *discrete geometric* (DG) setting. The equilibrium strategies for the DG setting are unknown, and the fairness implications of the protocols for the DG setting are unclear. A key challenge is that finding equilibrium in the DG setting is NP-complete (Garey et al., 1976), and it is computationally intensive to find equilibrium even for small input sizes. For example, an implicit enumeration to compute the equilibrium in the bisection protocol can take over two hours on a standard computer for a small grid with four districts and 36 units (Ludden et al., 2023). In contrast, the smallest congressional districting instance with four districts is Utah, with 716 census tracts. Hence, there is a need for a computationally efficient approach to execute the game protocols in practice.

This chapter implements the bisection and ICYF protocols in the practical DG setting. Since finding equilibrium is intractable for large instances, heuristic strategies are presented, guided by the equilibrium strategies in the CN setting. For the ICYF protocol in the CN setting, in every round, each player attempts to maximize the number of pieces they win, while maximally "packing" their opponent's voters into the rest of the pieces. For the DG setting, the drawing action is modeled as a mixed integer program (MIP) that optimizes this dual objective. This MIP ensures the contiguity of each district's geographical units and the population balance among the districts. For the bisection protocol in the CN setting, in every round, the bisecting player allocates a certain amount of their vote-share into the two pieces in anticipation that this allocation will yield the optimal number of winning districts in the end. For the DG setting, we use this allocation of vote-shares, and the bisection is formulated as an MIP to ensure that the two parts of the bisection conform to this allocation of vote-shares. Besides contiguity and population balance, this MIP incorporates a compactness objective to promote well-rounded districts. Furthermore, an investigation of the heuristic strategy for small grid examples reveals that the seat-share for the two parties mirrors the seat-shares under equilibrium play (computed using implicit enumeration in Ludden et al., 2023). Motivated by the closeness of the heuristic strategy to equilibrium in the small examples, the heuristics are then applied to practical instances of congressional districting in 18 states.

The rest of this chapter is organized as follows. Section 4.2 introduces the two protocols and the notations used in the rest of this chapter. Section 4.3 presents the heuristic for the two protocols, and compares the heuristic vs. optimal play for small grid examples. Section 4.4 applies the bisection and ICYF heuristics to a congressional districting instance in Iowa with a detailed case study on the heuristic strategies. Section 4.5 shows the scalability of the bisection heuristic by applying it to larger congressional districting instances in 18 states. Finally, Section 4.6 provides concluding remarks.

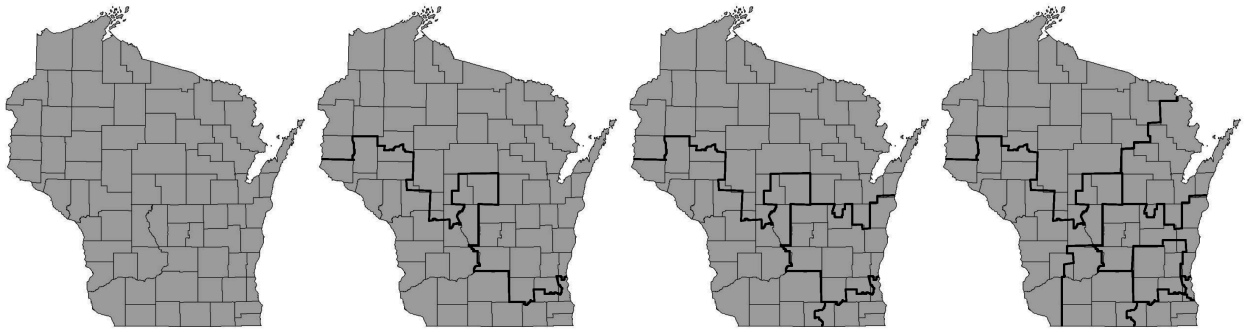
4.2 Background

The bisection and ICYF protocols are sequential games between two players (labeled 1 and 2 in the rest of this chapter). The overall objective for each player is to maximize the number of districts they win. This section presents the notations and algorithmic details for the bisection (Section 4.2.1) and ICYF (Section 4.2.2) protocols.

4.2.1 The Bisection Protocol

The bisection protocol (Ludden et al., 2023) is a districting game with two players. For a state with K districts, player 1 first partitions the state into two pieces with sufficient populations to comprise $\lfloor K/2 \rfloor$ and $\lceil K/2 \rceil$ districts. Player 2 then divides each of these in half (up to rounding). Player 1 then divides each of the

four pieces created by player 2 in half (up to rounding), and so on. A piece with an appropriate population for a single district is a finalized district, and the process terminates when all pieces are finalized districts. Figure 4.1 demonstrates how the eight congressional districts of Wisconsin for 2013–2022 could be obtained via the bisection protocol. A similar recursive bisection approach has been applied to graph partitioning before but with a single agent making each bisection (Bichot and Siarry, 2013, §1.9). The bisection protocol considered in this chapter can be viewed as a game-theoretic variation of this canonical recursive bisection algorithm with districting-focused objectives. Algorithm 4.1 describes the bisection protocol at a high level and is similar to Algorithm 1.2 of Bichot and Siarry (2013).



(a) WI counties (b) Player 1 bisects (c) Player 2 bisects (d) Player 1 bisects

Figure 4.1 The bisection protocol applied to Wisconsin.

Algorithm 4.1: Bisection protocol for partitioning a graph into a set $\mathcal{D}$ of K districts

1 **Algorithm** BisectionProtocol (graph $G = (V, E)$, number of districts $K \in \mathbb{Z}^+$)
2 $\quad \mathcal{D} \leftarrow$ RecursiveBisect (G, K, 1)
3 $\quad$ **return** $\mathcal{D}$

4 **Procedure** RecursiveBisect (subgraph H, number of districts m, player $\delta \in \{1, 2\}$)
5 $\quad$ **if** $m = 1$ **then**
6 $\quad\quad$ **return** $\{V(H)\}$
7 $\quad a \leftarrow \lfloor m/2 \rfloor$
8 $\quad (V_1, V_2) \leftarrow$ Player δ bisects H into connected components of weights a and $m - a$
9 $\quad \delta \leftarrow 3 - \delta$ `// Switch players`
10 $\quad \mathcal{D}_1 \leftarrow$ RecursiveBisect ($H[V_1]$, a, δ)
11 $\quad \mathcal{D}_2 \leftarrow$ RecursiveBisect ($H[V_2]$, $m - a$, δ)
12 $\quad$ **return** $\mathcal{D}_1 \cup \mathcal{D}_2$

Recall from Chapter 2 that in a real-world districting instance, the input graph G has a vertex for each indivisible geographic unit (e.g., census tract) and includes an edge between two units if they have a (nontrivial) shared border. In every round of bisection, the drawing player recieves a subgraph H of G which is to be divided into two parts with sizes $a = \lfloor m/2 \rfloor$ and $b = m - a$, where $m \in [K]$ is the total number of districts that can be drawn from that region. In the first round, $H = G$ and $m = K$. Vertices are weighted with the populations of the corresponding units, so the player-controlled bisection step in Line 8 of Algorithm 4.1 must produce parts with populations approximately proportional to the district counts (i.e., V_1 comprises

approximately an a/m fraction of the total population in H). Section 4.3.1 presents a heuristic method to execute the bisection in Line 8.

4.2.2 I-cut-you-freeze

Algorithm 4.2: I-cut-you-freeze protocol for partitioning a graph into a set $\mathcal{D}$ of K districts, adapted from Pegden et al., 2017

1 **Algorithm** IcyfProtocol (graph $G = (V, E)$, number of districts $K \in \mathbb{Z}^+$)
2 $\quad \mathcal{D} \leftarrow$ RecursiveIcyf (G, K, 1)
3 $\quad$ **return** $\mathcal{D}$

4 **Procedure** RecursiveIcyf (subgraph H, number of districts m, drawing player $\delta \in \{1, 2\}$)
5 $\quad$ **if** $m = 1$ **then**
6 $\quad\quad$ **return** $\{V(H)\}$
7 $\quad \psi \leftarrow 3 - \delta$ `// Other player index`
8 $\quad \{V_i\}_{i=1}^{m} \leftarrow$ player δ partitions H into m population-balanced districts
9 $\quad i^* \leftarrow$ player ψ chooses index of district to freeze
10 $\quad \delta \leftarrow 3 - \delta$ `// Players switch roles for next round`
11 $\quad \mathcal{D} \leftarrow$ RecursiveIcyf ($H[V(H) \setminus V_{i^*}]$, $m - 1$, δ)
12 $\quad$ **return** $\mathcal{D} \cup \{V_{i^*}\}$

The I-cut-you-freeze (ICYF) protocol is also a districting game with two players, where players alternate drawing and freezing actions (Pegden et al., 2017). The drawing player in the first round draws a district plan with K districts in the state, and the freezing player picks one of those districts to freeze. Then, the players switch roles, and the new drawing player draws $K - 1$ districts from the remaining region, and so on. This continues for $K - 1$ rounds; in the last round, the drawing player draws the last 2 districts. Algorithm 4.2 describes the ICYF protocol at a high level. Section 4.3.3 provides heuristic strategies for both the drawing (Line 8) and the freezing (Line 9) players.

4.3 Heuristics

For small instances, one may efficiently compute equilibrium bisection strategies by implicit enumeration of the game tree. But when the input graph is too large for exact equilibrium computation, how should the players bisect on their turns? This section provides heuristic strategies for both the bisection (Section 4.3.1) and ICYF (Section 4.3.3) protocols. Section 4.3.2 compares the strategies and outcomes of the bisection heuristic applied to small grid instances against the exact equilibria from Ludden et al., 2023.

4.3.1 Bisection Heuristic

The bisection heuristic for the DG setting is motivated by the game tree of the bisection protocol in the CN setting in Ludden et al., 2023. If a player had unlimited time and computational resources, they could build the entire DG game tree, then pick the bisection that gives them the most districts in each of their turns, assuming their opponent's play is optimal. As shown in Ludden et al., 2023, implicit enumeration of the game tree can take over two hours for a small 6×6 grid. As a result, even the smallest practical instances are unlikely to be tractable using an implicit enumeration approach.

Rather than attempting to look ahead at future rounds, the bisection heuristic replicates the vote-share allocation strategy in the CN setting, which essentially packs the opponents voters' into one piece, and cracks their voters in the other piece. The choice to imitate the CN vote-share allocation is motivated by the similarity between the CN and DG optimal first-round bisections in small grid instances (Ludden et al., 2023).

Furthermore, preliminary investigations (see Appendix C.2 in the e-companion) revealed that when the physical shapes of the bisected pieces are not *compact*, or round-shaped, there may not be feasible bisections in future rounds. To prevent such infeasibility, the bisection heuristic finds the most compact bisection that achieves the target vote-shares from the CN setting in the two pieces. There are many ways to quantify compactness (Duchin and Tenner, 2018); this chapter chooses to minimize the number of *cut edges*, that is, edges whose end points are in different pieces. The cut-edge compactness measure is a popular discrete compactness score that agrees well with visual intuition (Validi and Buchanan, 2022; DeFord et al., 2021a). Note the compactness objective might unintentionally bias the pieces in favor of a party whose spatial voter distribution benefits from compact pieces (Chen and Rodden, 2013).

The remainder of this section explains how the heuristic determines the target vote-shares and finds a bisection. In a given call to RecursiveBisect in Algorithm 4.1, player $\delta \in \{1, 2\}$ receives a subgraph $H \subseteq G$ to be divided into two parts totaling $m \in [K]$ districts (e.g., in the first round, $\delta = 1$, $H = G$, and $m = K$.) Recall from Chapter 2 that each unit $i \in V(H)$ has population p_i and player voters p_i^r for $r \in [2]$ (Note that player indices 1 and 2 replace party indices A and B in Chapter 2). Let $\psi = 3 - \delta$ denote the other player's index, and let $P^\delta(H) = m \cdot \left(\sum_{i \in V(H)} p_i^\delta\right) / \left(\sum_{i \in V(H)} (p_i^\delta + p_i^\psi)\right)$ denote the overall vote-share in H for player δ. Then, the equilibrium strategy in the CN setting gives the number of possible districts $j^* \in [m]$ that can be won for this overall vote-share in H for player δ. It also gives the number of districts $k^* \in [j^*]$ and $j^* - k^*$ that can be won in the pieces with sizes a and b, respectively. To achieve this, the drawing player must allocate vote-shares of $\omega_\delta^a, \omega_\delta^b \in [0, 1]$, respectively, in the pieces with sizes a and b. This vote-share allocation for the two pieces is the key input to the bisection heuristic in the DG setting.

If player δ begins with an overall vote-share of at least $(a\omega_\delta^a + b\omega_\delta^b)/(a + b)$, then splitting the vote-share into ω_δ^a and ω_δ^b is always feasible in the CN setting and yields j^* districts. In the DG setting, however, such a split may not be feasible. To address this, a feasibility mixed-integer program (MIP) named Bisection is designed to check whether there exists a feasible bisection with at least ω_δ^a in the side of size a and ω_δ^b in the side of size b. If the MIP is infeasible, then j^* is iteratively decremented by 1 until the MIP yields a feasible solution. This assumes that player δ will, upon realizing they cannot win j^* districts, attempt to win one fewer. Algorithm 4.3 describes this bisection heuristic to be used in Line 8 of Algorithm 4.1.

Algorithm 4.3: Bisection protocol for the discrete geometric setting implementing Line 9 of Algorithm 4.1

```
1 Algorithm Bisect (H, m, δ)
2     Compute j*, k* using CN setting (Ludden et al., 2023)
3     while Bisection(H, m, j*, k*, δ) is infeasible do
4         j* ← j* − 1
5         Recompute k* using CN setting (Ludden et al., 2023)
6     return feasible solution of Bisection(H, m, j*, k*, δ) as vertex sets V1, V2
```

The Bisection(H, m, j^*, k^*, δ) feasibility MIP is now defined. Let $\overline{P}_a := (a/m) \cdot (\sum_{i \in V(H)} p_i)$ be the ideal total population of the piece of size a (i.e., $p(V_1)$). To support flow-based contiguity constraints adapted from Shirabe, 2009, let $\overrightarrow{H} = (V(H), \overrightarrow{E}(H))$ be a bidirected version of H (i.e., $\overrightarrow{E} = \bigcup_{uv \in E} \{\vec{uv}, \vec{vu}\}$), and let

$N_H(i) := \{j \in V(H) : ij \in E(H)\}$ be the neighbors of unit i in H.

The variables for the MIP are

- the assignment indicators $x_i = 1$ if and only if $i \in V(H)$ is assigned to V_1 (otherwise, i is in V_2),
- the cut edge indicators $y_{ij} = 1$ for $ij \in E(H)$ if and only if exactly one endpoint is assigned to V_1,
- the flows $g^k_{i,j}$ = amount of directional flow on edge $ij \in \overrightarrow{E}(H)$ for piece $k \in \{1, 2\}$ (adapted from Shirabe, 2009) and
- the sink indicators $\beta_{i,k} = 1$ if and only if $i \in V(H)$ is the sink for piece $k \in \{1, 2\}$ (adapted from Shirabe, 2009).

The x variables fully determine $V_1 = \{i \in V(H) : x_i = 1\}$ and $V_2 = \{i \in V(H) : x_i = 0\}$, which are returned at the end of Algorithm 4.3 if a feasible solution is found. The y variables are used in formulating the compactness objective which minimizes the number of cut edges. The flow and sink variables enforce contiguity by directing at least one unit of flow from each unit in a district to its district's sink through other units in the district; see Shirabe, 2009 for details. The program Bisection(H, m, j^*, k^*, δ) is then

$$\text{minimize} \quad \sum_{ij \in E(H)} y_{i,j} \tag{4.1a}$$

$$\text{subject to} \quad (1-\tau)\overline{P}_a \le \sum_{i \in V(H)} p_i x_i \le (1+\tau)\overline{P}_a, \tag{4.1b}$$

$$\sum_{i \in V(H)} p^\delta_i x_i \ge \omega^a_\delta \sum_{i \in V(H)} (p^\delta_i + p^\psi_i) x_i, \tag{4.1c}$$

$$\sum_{i \in V(H)} p^\delta_i (1 - x_i) \ge \omega^b_\delta \sum_{i \in V(H)} (p^\delta_i + p^\psi_i)(1 - x_i), \tag{4.1d}$$

$$\sum_{i \in V(H)} \beta_{i,k} = 1 \qquad \forall k \in \{1, 2\}, \tag{4.1e}$$

$$\beta_{i,1} \le x_i \qquad \forall i \in V(H), \tag{4.1f}$$

$$\beta_{i,2} \le 1 - x_i \qquad \forall i \in V(H), \tag{4.1g}$$

$$\sum_{j \in N_H(i)} (g^1_{i,j} - g^1_{j,i}) \ge x_i - |V(H)|\,\beta_{i,1} \qquad \forall i \in V(H), \tag{4.1h}$$

$$\sum_{j \in N_H(i)} (g^2_{i,j} - g^2_{j,i}) \ge (1 - x_i) - |V(H)|\,\beta_{i,2} \qquad \forall i \in V(H), \tag{4.1i}$$

$$\sum_{j \in N_H(i)} g^1_{i,j} \le |V(H)|\,x_i \qquad \forall i \in V(H), \tag{4.1j}$$

$$\sum_{j \in N_H(i)} g^2_{i,j} \le |V(H)|\,(1 - x_i) \qquad \forall i \in V(H), \tag{4.1k}$$

$$y_{ij} \le x_i + x_j \qquad \forall ij \in E(H), \tag{4.1l}$$

$$y_{ij} \le 2 - x_i - x_j \qquad \forall ij \in E(H), \tag{4.1m}$$

$$y_{ij} \ge x_i - x_j \qquad \forall ij \in E(H), \tag{4.1n}$$

$$y_{ij} \ge -x_i + x_j \qquad \forall ij \in E(H), \tag{4.1o}$$

$$x_i, y_{ij}, \beta_{i,k} \in \{0, 1\} \qquad \forall i \in V(H), k \in \{1, 2\}, j \in N_H(i), \tag{4.1p}$$

$$g^k_{i,j} \ge 0 \qquad \forall i \in V(H), k \in \{1, 2\}, j \in N_H(i). \tag{4.1q}$$

Note that the indices j and k used in defining the MIP are not related to the variables j^* and k^* introduced above to define the target seat-shares. The objective in (4.1a) minimizes the number of cut edges. Constraints (4.1b) ensure population balance in the smaller (or equal size) side, V_1. Note that constraints (4.1b) generalize the classical partition problem, which is NP-complete (Garey et al., 1976). Furthermore, constraints (4.1b) also ensure population balance in the larger (or equal size) side, V_2, because the absolute population deviations in V_1 and V_2 are equal in magnitude but are divided by a and $b \geq a$ to obtain the relative deviations. Constraints (4.1c) and (4.1d) ensure that player δ has at least ω_δ^a vote-share in V_1 and at least ω_δ^b vote-share in V_2. Constraints (4.1e)-(4.1k) are flow-balance constraints that ensure that each piece is contiguous, adapted from Shirabe, 2009. Constraints (4.1e) ensure that each piece has a single unit serving as a sink for flow, and constraints (4.1f)-(4.1g) ensure that a unit i can serve as the sink of piece k only if i is assigned to piece k. Constraints (4.1h)-(4.1i) ensure that for every unit i assigned to piece k, the net piece-k-flow leaving i is at least 1 if i is not chosen as sink; the net piece-k-flow leaving i is unbounded if i is chosen as a sink. Constraints (4.1j)-(4.1k) ensure that a unit i sends out flow in piece k only if i is assigned to piece k. Constraints (4.1l)-(4.1o) define the cut edge indicator $y_{ij} = x_i \oplus x_j$ for each edge $ij \in E(H)$ with respect to the assignment variables. Constraints (4.1p)-(4.1q) characterize the binary and continuous variables.

4.3.2 Comparing Heuristic and Optimal Bisection on Small Grids

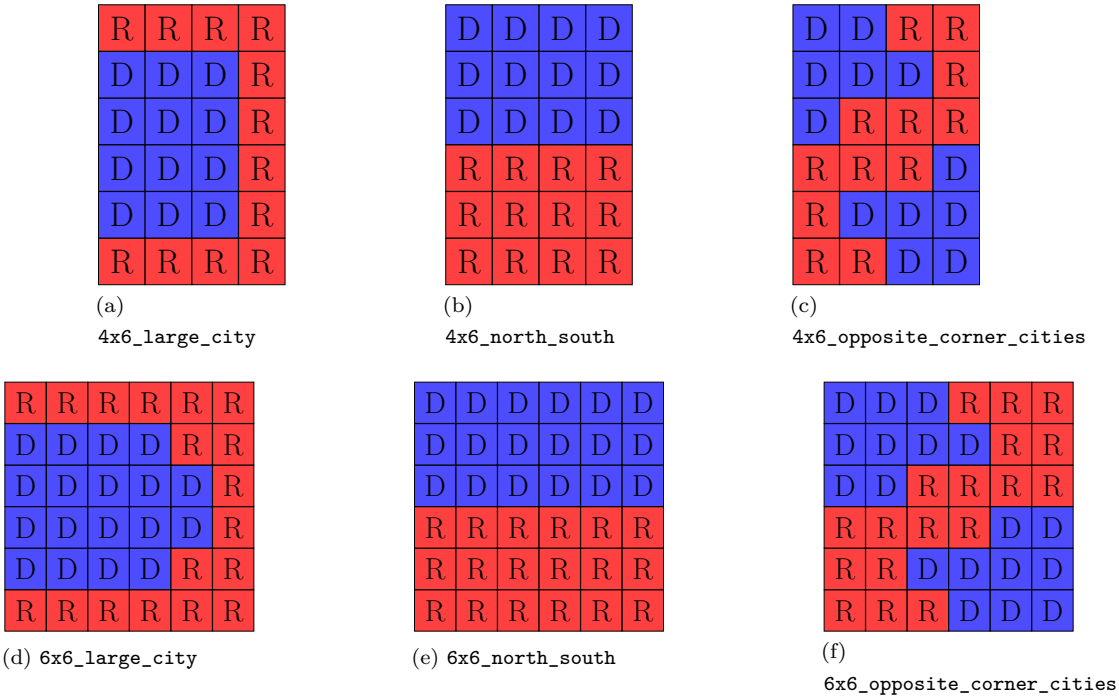


Figure 4.2: Examples of two-party spatial distributions of voters for small grid graphs (Ludden et al., 2023).

The bisection heuristic in Section 4.3.1 offers a practical, but not necessarily optimal, method for each player to bisect in their round. This section explores the extent to which the heuristic differs from optimal play on small grid instances. Figure 4.2 presents 4×6 and 6×6 grid graphs with three hypothetical spatial voter distributions: one large city on the western border, a symmetric north/south split, and two medium cities in opposite corners. Each unit has the same population and is fully for one party or the other. All six instances

have 50% vote-share. These instances into $K = 4$ or $K = 6$ districts with perfect population balance, which requires two or three rounds of bisection, respectively.

The bisection heuristic is applied to the six small grid instances. Table 4.1 reports the utilities when either party draws first, and when the number of districts is $K = 4$ or $K = 6$. Here, a table entry of "3D, 1T, 2R" indicates three wins for party D, one tie, and two wins for party R. The heuristic utilities align with the optimal utilities in Ludden et al., 2023, assuming that ties are awarded to the first player. Among the 6×6 instances, when $K = 6$, the heuristic utilities align with the optimal utilities in all three instances; when $K = 4$, the optimal strategy awards the first player one extra district. Among the 4×6 instances, when $K = 6$, the heuristic utilities align with the optimal utilities in all but one instance (`4x6_large_city`); when $K = 4$, the heuristic and optimal utilities differ. These differences in the smaller instances arise due to the local versus global decision-making when applying a heuristic versus the optimal strategy.

Table 4.1: Utilities from applying the bisection heuristic on small grid instances.

Spatial Voter Distribution	$K = 4$ Districts		$K = 6$ Districts	
	D first	R first	D first	R first
`4x6_large_city`	4T	4T	1D, 4T, 1R	2R, 2T, 2D
`4x6_north_south`	2D, 2R	2R, 2D	2D, 2T, 2R	2R, 2T, 2D
`4x6_opposite_corner_cities`	4T	4T	1D, 4T, 1R	1R, 4T, 1D
`6x6_large_city`	2D, 2R	2R, 2D	4D, 2R	2R, 2T, 2D
`6x6_north_south`	2D, 2R	2R, 2D	2D, 2T, 2R	2R, 2T, 2D
`6x6_opposite_corner_cities`	2D, 2R	2R, 2D	2D, 2T, 2R	2R, 2T, 2D

The two strategies also differ in the computational time required to run the protocol. The implicit enumeration technique in Ludden et al., 2023 takes 1–11 seconds for the 4×6 instances and 49–8,379 seconds for the 6×6 instances. The bisection heuristic, on the other hand, takes less than 1 second to solve *all* of the small grid instances. As will be shown in Sections 4.4 and 4.5, the bisection heuristic can therefore complete the protocol for larger instances.

4.3.3 I-Cut-You-Freeze Heuristics

The heuristic strategies for ICYF in the DG setting are based on the optimal strategies presented in Pegden et al. (2017) for ICYF in the CN setting. Algorithm 4.2 presents ICYF in the DG setting with only the drawing step of Line 8 and the freezing step of Line 9 left undetermined. Note that a heuristic implementation of ICYF requires both a freezing heuristic and a drawing heuristic, whereas the heuristic implementation of the bisection protocol needs only a drawing heuristic.

In the CN setting, the optimal freezing strategy is to choose a district the freezing player wins by the least margin; if no such district exists, the freezing player should choose a district in which their opponent wins by the greatest margin (Pegden et al., 2017). Since the freezing strategy only depends on victory margins, it can be directly applied in the DG setting. Similarly, the ICYF drawing strategy may be adapted from the CN setting. In the CN setting, the drawing player should maximize the number of districts they win while maximally packing their opponent's voters in the districts they lose (Pegden et al., 2017). The DG setting inherently limits packing, but the same objectives can be pursued by formulating and solving a MIP. Applying this drawing strategy for ICYF is a heuristic rather than an exact algorithm because it does not attempt to look ahead to future rounds. The structure of the subgraph in later rounds, which is partially determined by the drawing and freezing of the current round, may prevent partitions that mirror the optimal

strategies for ICYF in the CN setting. But among the class of restricted heuristics that do not explore further down the game tree, this drawing heuristic for ICYF is a reasonable approach that captures the essential packing and cracking tactics from the CN setting.

The ICYF MIP for the heuristic drawing strategy used by player δ in Line 8 is constructed from the parameters H, m, and δ of RecursiveIcyf. Define ψ, $N_H(i)$, and $\overrightarrow{H}$ as in Section 4.3.1. The variables for the ICYF MIP are

- the assignment indicators $x_{i,k} = 1$ if and only if $i \in V(H)$ is assigned to district $k \in [m]$,
- the district winner indicators $z_k = 1$ if and only if player δ wins district $k \in [m]$,
- the opponent's unit winner indicators $u_{i,k} = 1$ if and only if $i \in V(H)$ is assigned to district $k \in [m]$ and player ψ wins district k,
- the opponent's winning vote-shares divided by m,

$$q_k = \begin{cases} \frac{1}{m} \cdot (\text{player } \psi \text{ vote-share in district } k) & \text{if player } \psi \text{ wins district } k \in [m] \\ 1 & \text{otherwise,} \end{cases}$$

- the minimum of all the q_k values, $q = \min\{q_k : k \in [m]\}$ (which is 1 if player δ wins every district),
- the flows $g^k_{i,j}$ = amount of directional flow on edge $(i,j) \in \overrightarrow{E}(H)$ for district $k \in [m]$ (adapted from Shirabe, 2009), and
- the sink indicators $\beta_{i,k} = 1$ if and only if $i \in V(H)$ is the sink for district $k \in [m]$ (adapted from Shirabe, 2009).

In defining the winner indicators z, a tie is awarded to the drawing player. The flow and sink variables enforce contiguity by directing at least one unit of flow from each unit in a district to its district's sink through other units in the district; see Shirabe, 2009 for details. The MIP ICYF(H, m, δ) is then

$$\text{maximize} \quad \left(\sum_{k\in[m]} z_k\right) + q \tag{4.2a}$$

$$\text{subject to} \quad \sum_{k\in[m]} x_{i,k} = 1 \qquad \forall i \in V(H), \tag{4.2b}$$

$$(1-\tau)\overline{P} \le \sum_{i\in V(H)} p_i x_{i,k} \le (1+\tau)\overline{P} \qquad \forall k \in [m], \tag{4.2c}$$

$$-M(1-z_k) \le \sum_{i\in V(H)} \left(p_i^{\delta} - p_i^{\psi}\right) x_{i,k} \qquad \forall k \in [m], \tag{4.2d}$$

$$\sum_{i\in V(H)} \left(p_i^{\delta} - p_i^{\psi}\right) x_{i,k} \le M z_k \qquad \forall k \in [m], \tag{4.2e}$$

$$u_{i,k} \le x_{i,k} \qquad \forall i \in V(H), k \in [m], \tag{4.2f}$$

$$u_{i,k} \le 1 - z_k \qquad \forall i \in V(H), k \in [m], \tag{4.2g}$$

$$u_{i,k} \ge x_{i,k} - z_k \qquad \forall i \in V(H), k \in [m], \tag{4.2h}$$

$$q_k = z_k + \frac{1}{m\overline{P}} \sum_{i\in V(H)} p_i^{\psi} u_{i,k} \qquad \forall k \in [m], \tag{4.2i}$$

$$q \leq q_k \qquad \forall k \in [m], \tag{4.2j}$$

$$\sum_{i \in V(H)} \beta_{i,k} = 1 \qquad \forall k \in [m], \tag{4.2k}$$

$$\beta_{i,k} \leq x_{i,k} \qquad \forall i \in V(H), k \in [m], \tag{4.2l}$$

$$\sum_{j \in N_H(i)} (g^k_{i,j} - g^k_{j,i}) \geq x_{i,k} - |V(H)|\, \beta_{i,k} \forall i \in V(H), k \in [m], \tag{4.2m}$$

$$\sum_{j \in N_H(i)} g^k_{i,j} \leq |V(H)|\, x_{i,k} \qquad \forall i \in V(H), k \in [m], \tag{4.2n}$$

$$x_{i,k}, z_k, u_{i,k}, \beta_{i,k} \in \{0,1\} \qquad \forall i \in V(H), k \in [m], \tag{4.2o}$$

$$q, q_k \in [0,1] \qquad \forall k \in [m], \tag{4.2p}$$

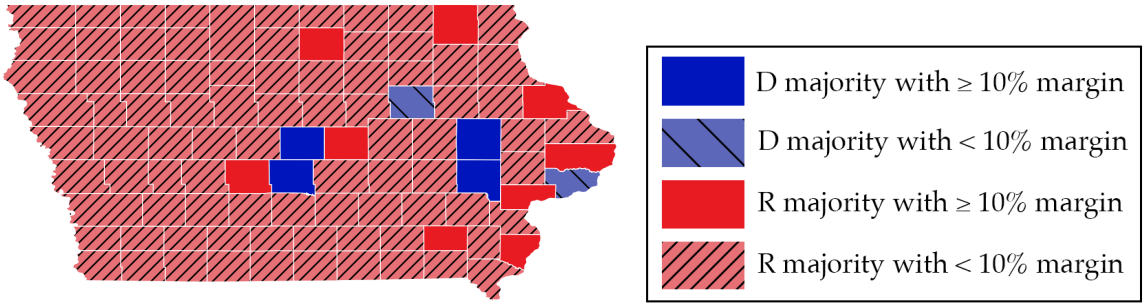
$$g^k_{i,j} \geq 0 \qquad \forall i,j \in V(H), k \in [m]. \tag{4.2q}$$

The objective in (4.2a) maximizes the number of districts won by player δ with a secondary objective of maximizing the minimum margin among the districts won by player ψ. The minimum margin is scaled down by m to ensure q is fractional, and therefore a secondary objective to the integral sum of z_k indicators, whenever player ψ wins at least one district. These two objectives align with those of the drawing player in the CN setting. Constraints (4.2b) assign each unit to exactly one district. Constraints (4.2c) ensure that each district population lies within the permissible range. Constraints (4.2d)-(4.2e) relate the binary variables x and z for some $M > \max\{0, \sum_{i \in V}(p^\delta_i - p^\psi_i)\}$. Constraints (4.2f)-(4.2h) linearize the quadratic constraints given by $u_{i,k} = x_{i,k}(1 - z_k)$. When there is a tie in district k, constraints (4.2f)-(4.2h) allow z_k to be either 0 or 1. In this case, maximizing the objective function in (4.2a) sets z_k to be 1, rewarding the district to the drawing player. Constraints (4.2i) use the vote counts and the $u_{i,k}$ indicator variables to define the victory margin for each district that player ψ wins, dividing by the total district population $m\overline{P}$ rather than the total number of voters given by $\sum_{i \in V(H)}(p^\delta_i + p^\psi_i)x_{i,k}$ in each district $k \in [m]$ to keep the constraints linear. Dividing by $m\overline{P}$ rather than simply $\overline{P}$ prevents q_k from exceeding 1 in the case when district k has population greater than $\overline{P}$. Scaling victory margins down by a factor of m does not change their ordering, so maximizing the minimum scaled-down victory margin is equivalent to maximizing the minimum original victory margin. Constraints (4.2j) define the (scaled-down) minimum margin among the districts won by player ψ. Constraints (4.2k)-(4.2n) are flow-balance constraints that ensure each district is contiguous, adapted from Shirabe, 2009. Constraints (4.2k) ensure each district has a single unit serving as a sink for flow, and constraints (4.2l) ensure that unit i can serve as the sink of district k only if i is assigned to district k. Constraints (4.2m) ensure that for every unit i assigned to district k, the net district-k-flow leaving i is at least 1 if i is not chosen as sink; the net district-k-flow leaving i is unbounded if i is chosen as a sink. Constraints (4.2n) ensure that unit i sends out district-k-flow only if i is assigned to district k. Hence, if district k is noncontiguous, these constraints ensure that the flow from every unit assigned to k will not reach the sink of district k. Constraints (4.2o)-(4.2q) enforce the binary or continuous bounds on the variables.

4.4 Computational Results: Iowa Case Study

This section presents computational results for the application of the discrete protocol implementations from Sections 4.3.1 and 4.3.3 to the state of Iowa. Iowa has only four congressional districts and requires that none of its 99 counties are split across districts, making it an ideal state for a small-scale case study on a practical

districting scenario. The vote counts for the two major parties, Democratic (D) and Republican (R), are derived by averaging the 2016 and 2020 presidential election results disaggregated from voting precincts to each county (MIT Election Data and Science Lab, 2020). Based on this dataset, the overall vote-share for R in Iowa is 55%. Figure 4.3 categorizes the vote counts in each county, giving an overview of the political geography of Iowa. The population balance threshold τ is set to 1%. All computations were performed on a 3.1 GHz Intel Core i5-2400 CPU machine using a single core. The mixed integer programs were solved using the IBM CPLEX Optimizer 12.10.



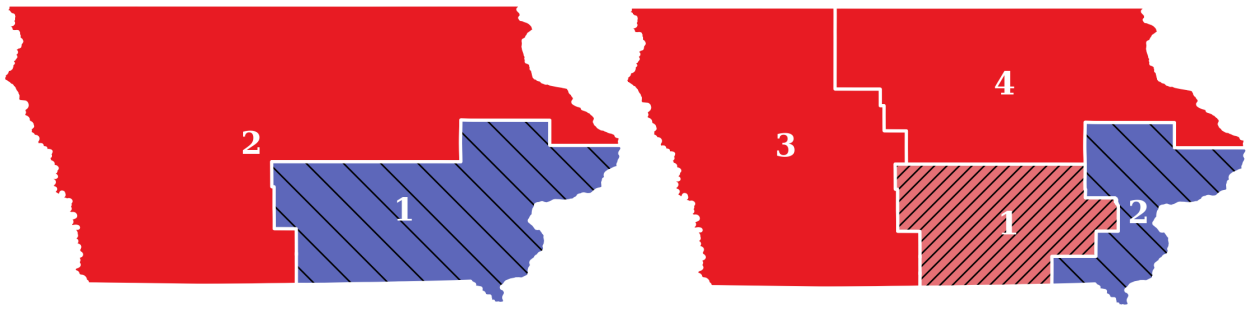
(a) County-level party support for D and R (b) Legend

Figure 4.3 County-level party support based on 2016 and 2020 presidential elections in Iowa.

The two districting protocols are applied to Iowa twice: once for each player (D or R) taking the first turn. Further, the two protocols are compared with respect to fairness metrics and computational times. Note the results presented in this section are obtained using the particular implementation of the heuristics from Sections 4.3.1 and 4.3.3 for both players, and may not reflect the (unknown) optimal play for the two protocols.

Bisection.

The bisection protocol (Algorithm 4.1) is applied to Iowa congressional districting using the MIP-based bisection heuristic (Algorithm 4.3) to divide each set of counties into two pieces (Line 8 of Algorithm 4.1). With $K = 4$ districts, there are two rounds, that is, two levels of the recursion tree.



(a) First round: D bisects the state (b) Second round: R bisects each piece

Figure 4.4 Bisection protocol when D bisects first.

Figure 4.4 depicts the results from applying the bisection protocol when D draws the first round and R draws the second round. In the final district plan (Figure 4.4b), D wins one out of the four districts. The

vote-shares for D in the four districts are, in decreasing order, 0.54, 0.49, 0.45, and 0.34. This district plan has an efficiency gap of 15.49% in favor of R, and a partisan asymmetry of 2.84%. The total CPU time to execute the protocol is 3.26 seconds.

When D draws the first round, their overall vote-share of 45% (1.8) gives two wins in the CN setting, so they try to win two districts from one piece of the bisection. To achieve this, D first attempts to allocate at least 75% (1.5) vote-share in the first piece, but the MIP is infeasible. Then, D reduces the target vote-share allocation to 50% (1) in the first piece, which yields a feasible bisection. The resulting pieces, labeled 1 and 2 in Figure 4.4a, have D vote-shares 51% (1.02) and 39% (0.78). Next, R divides the two pieces into four smaller pieces in the next round by allocating 50% (1) and 0% (0) vote shares in each piece's bisection. As illustrated by Figure 4.4b, R first bisects piece 1 into a win for R (district 1) and a win for D (district 2), then bisects piece 2 into two wins for R (districts 3 and 4). Overall, R receives a greater seat-share than what is expected from the CN thresholds.

(a) First round: R bisects the state

(b) Second round: D bisects each piece

Figure 4.5 Bisection protocol when R bisects first.

Swapping turn order, Figure 4.5 depicts the results from applying the bisection protocol when R draws the first round and D draws the second round. In the final district plan (Figure 4.5b), each party wins two districts. The district vote-shares for R are 0.68, 0.58, 0.47, and 0.46. This district plan has an efficiency gap of 9.57% in favor of D, and a partisan asymmetry of 2.27%. The total CPU time to execute the protocol is 2.60 seconds.

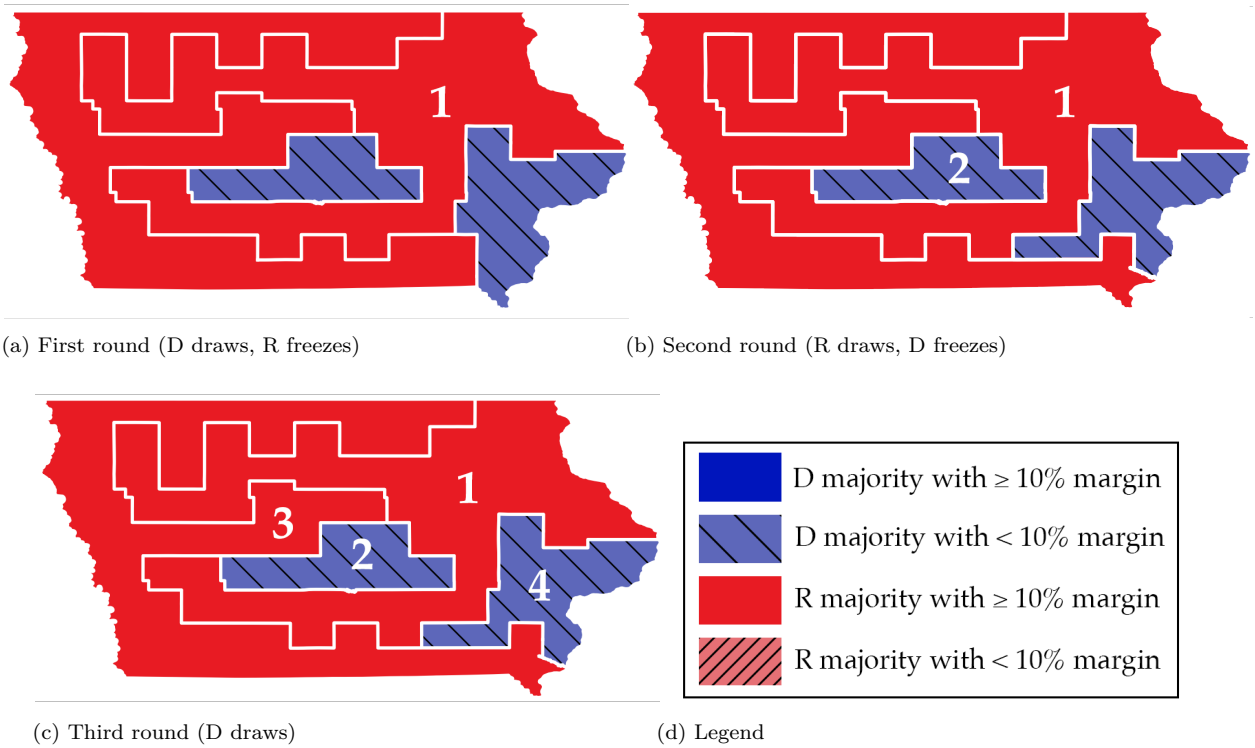
When R draws the first round, their overall vote-share of 55% (2.2) gives two wins in the CN setting, so they try to win two districts from one piece of the bisection. To achieve this, R first attempts to allocate 75% (1.5) vote-share in the first piece, but the MIP is infeasible. Then, R reduces the target vote-share allocation to 50% (1) in the first piece, which yields a feasible bisection. The resulting pieces, labeled 1 and 2 in Figure 4.5a, have R vote-shares 52% (1.04) and 57% (1.14). Next, D divides the two pieces into four smaller pieces in the next round by allocating 50% (1) and 0% (0) vote shares in each piece's bisection. As illustrated by Figure 4.5b, D first bisects piece 1 into a win (district 1) and a loss (district 2), then bisects piece 2 into a win (district 3) and a loss (district 4).

The DG setting bisection heuristic results differ in strategy and outcome from the outcome expected from the CN setting's thresholds. For the given overall vote-shares in Iowa, the CN thresholds indicate that the first player (D or R) can win two districts overall, both of which are from one piece of the first bisection. However, when D draws first, the final district plan only yields one D district. When R draws first, the final district plan yields two R districts, although they are spread across the two pieces. This discrepancy between the final outcome and what was expected is due to a combination of (i) the inherent differences between the

DG and CN settings, and (ii) the approximate nature of applying a heuristic when compared to optimal play.

I-Cut-You-Freeze.

The ICYF protocol as described in Algorithm 4.2 is applied to Iowa congressional districting, using the MIP-based ICYF heuristic presented in Section 4.3.3 to draw the districts in Line 8. For $K = 4$, there are three rounds of ICYF. Note that the third round of drawing essentially freezes the last two districts. In each round, a time limit of 1 hour is set for the MIP solver. If an optimal solution is not found within the time limit, the solver returns the best known solution.



(a) First round (D draws, R freezes)

(b) Second round (R draws, D freezes)

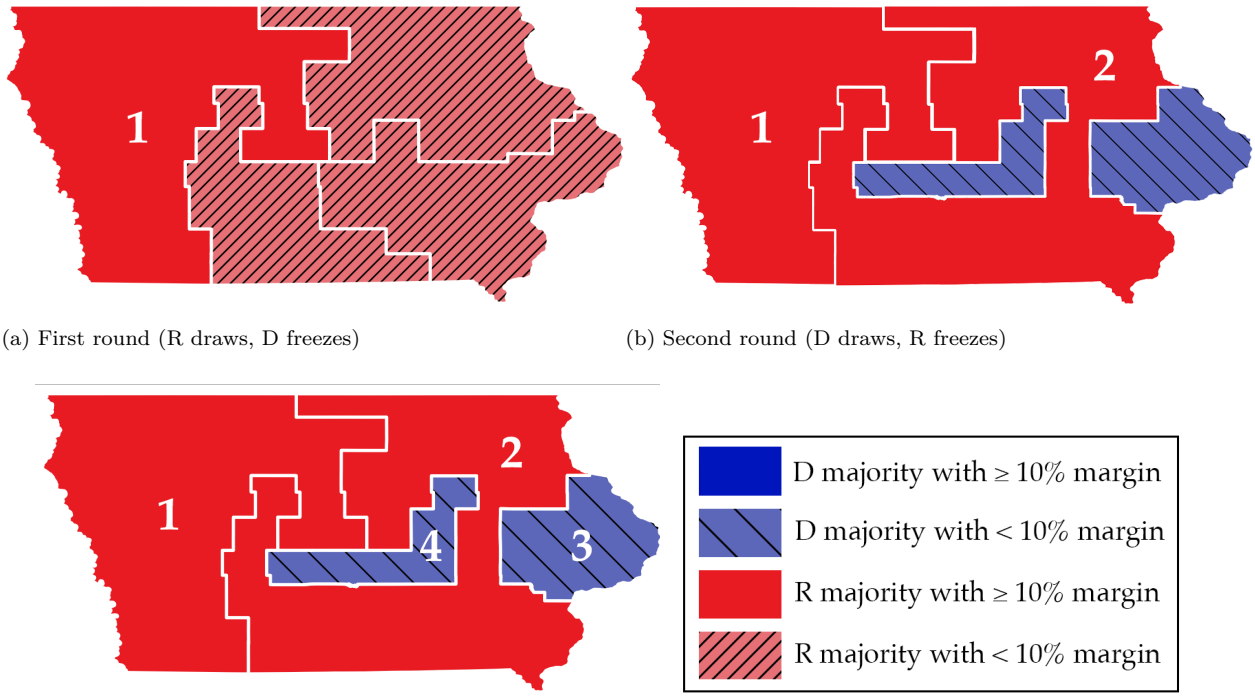
(c) Third round (D draws)

(d) Legend

Figure 4.6 I-Cut-You-Freeze protocol to draw four districts in Iowa when D draws the first round.

Figure 4.6 depicts heuristic play of the ICYF protocol when D draws first. In the final district plan (Figure 4.6c), D has district vote-shares 0.54, 0.54, 0.41, and 0.33, winning two districts. This district plan has an efficiency gap of 8.39% in favor of D, and a partisan asymmetry of 2.16%. The total computational time to execute the three rounds is 1 hour, 43 minutes.

In the first round, D draws two wins and packs R into two districts with margins 21.51% and 30.93%. Solving the MIP to produce this plan takes 1 hour, when it terminates after reaching the time limit. Following the CN freezing strategy, R freezes district 1, their least-margin win. In the second round, R redraws the remaining three districts, producing the plan shown in Figure 4.6b. Solving this MIP takes 43 minutes. In this plan, R wins one district and packs D into two districts with margins of 7.68% and 8.92%. Then D freezes district 2, their least-margin win. In the third and final round, D draws districts 3 and 4. The MIP solver terminates with optimality in 3 seconds. In the resulting district plan (Figure 4.6c), R wins district 3 and D wins district 4 with margins 31.42% and 8.92%, respectively.



(a) First round (R draws, D freezes)

(b) Second round (D draws, R freezes)

(c) Third round (R draws)

(d) Legend

Figure 4.7 I-Cut-You-Freeze protocol to draw four districts in Iowa when R draws the first round.

Swapping turn order, Figure 4.7 shows heuristic ICYF play when R draws first. In the final district plan (Figure 4.7c), each party wins two districts; the district vote-shares of R are, in decreasing order, 0.64, 0.62, 0.46, and 0.45. This district plan has an efficiency gap of 9.24% in favor of D, and a partisan asymmetry of 0.20%. The total computational time to execute the three rounds is 15 minutes and 35 seconds.

In the first round, R draws four wins and D freezes the win with maximum margin (Figure 4.7a). In the second round, D redraws the remaining three districts, producing the district plan shown in Figure 4.7b. In this plan, D wins two of the three districts and packs R in a district with margin 25.80%, which R freezes to form district 2. The remaining region comprises two disconnected districts, which R is forced to redraw in the third and final round, producing the final district plan (Figure 4.7c).

Comparing Bisection and I-Cut-You-Freeze.

The geographical distribution of each party's voters impacts the ability of each player to pack and crack the opponent's voters. In particular, Iowan D voters are generally clustered in a few urban counties, and Iowan R voters are generally spread across rural counties. As evidence of this clustering, D has a majority vote count in only 6 of the 99 counties in Iowa despite having 45% of the statewide vote-share.

The two protocols do not provide any advantage to the first player. In fact, bisection yields a second-player advantage when D draws first. Further, ICYF yields the same seat-share for the players when either player draws first. The asymmetric political geography may partially explain the difference in outcomes between the two protocols. Note that these observations are anecdotal to the Iowa instance and might not generalize to districting instances in other states.

Table 4.2 reports the two partisan fairness metrics defined in Chapter 2 — efficiency gap and partisan

asymmetry — for the district plan produced from each protocol with each player taking the first turn. The metrics for Iowa's 115th congressional districts (115th CD), used for 2017-2019, are also included. Recall that efficiency gap captures the difference in wasted votes between the two players, and (partisan) asymmetry is the total area between the seats-votes curves of the two players. Hence, smaller values of all three metrics indicate fairer districts. For the bisection protocol, the district plan obtained when R goes first produces fairer districts (in all three metrics) than the plan when D goes first. For the ICYF protocol, the district plan obtained when D goes first has a slightly smaller efficiency gap, although the district plan obtained when R goes first has a smaller partisan asymmetry.

Table 4.2: District plans obtained from the bisection protocol, I-Cut-You-Freeze protocol and Iowa's 115th congressional districts compared across two partisan fairness metrics.

Fairness metric	Bisection		I-Cut-You-Freeze		115th CD
	D first	R first	D first	R first	
Seat-share for D	0.25	0.5	0.5	0.5	0.25
Efficiency Gap	15.49%	9.57%	8.39%	9.24%	14.68%
Partisan Asymmetry	2.84%	2.27%	2.16%	0.20%	7.95%
Computational Time (s)	3	3	6,183	935	-

The execution of the heuristic bisection protocol is drastically faster than that of the heuristic ICYF protocol. The observed CPU time for dividing a region into two pieces, as in the bisection protocol, is significantly lower than for dividing it into three or more pieces, as in the ICYF protocol. Additionally, bisection involves $\lceil \log_2 K \rceil$ rounds, whereas ICYF requires $K - 1$ rounds. These computational advantages suggest the bisection heuristic may be more likely than the ICYF heuristic to be tractable for larger districting instances.

4.5 Computational Results: Larger Instances

The Iowa case study demonstrates the protocol heuristics can be efficiently applied to a practical instance and produce fairer district plans than Iowa's 115th congressional districts. The robustness and scalability of the bisection heuristic are examined by applying it to 18 congressional districting instances in U.S. states at the census-tract level. The smallest instance is Utah with 716 census tracts and 4 districts, and the largest is Missouri with $1,654$ census tracts and 8 districts. Among these states, six are D-leaning, that is, have a D voter majority. The most D-leaning state is Maryland (65.6% D voters) and the most R-leaning state is Oklahoma (68.1% R voters). The MIP-based heuristic bisection protocol is applied to each instance with $\tau = 1\%$ and a one-hour MIP solver time limit as in Section 4.4. The four-district states have two rounds of bisection, and the others have three.

The bisection heuristic produces feasible district plans for all 18 states. Table 4.3 reports the partisan fairness metrics (efficiency gap and partisan asymmetry) of the final district plans obtained when D bisects first, and Figure C.1 presents the resulting plans. Table 4.4 and Figure C.2 provide the same for when R bisects first. In 12 states, the seat-share does not depend on who goes first. Furthermore, four states (Arkansas, Mississippi, Iowa and South Carolina) give R a greater seat-share when D goes first, and two states (Alabama and Wisconsin) give R a lower seat-share when R goes first. This suggests the bisection heuristic gives a slight advantage (w.r.t. seat-share) to the second player by not allowing the first player to draw convoluted districts that restrict future bisection strategies.

Table 4.3: The bisection protocol (when D draws the first round) with the MIP-based heuristic (Section 4.3.1) is applied to U.S. states with 4 to 8 congressional districts at the census tract-level. Abbreviations include EG for efficiency gap, PA for partisan asymmetry, and CN for the continuous non-geometric setting. *Positive efficiency gap is in favor of R; negative is in favor of D. Superscripts ^{T}, ^{TT}, and ^{TTT} indicate one, two, or three iterations of the MIP solver reached the one-hour time limit.

State	Number of census tracts	Number of districts	Overall R vote-share	Time taken (s)	R seat-share	EG*	PA	CN R seat-share
Utah	716	4	61.3%	615	75.0%	1.9%	4.8%	50.0%
Nevada	778	4	48.7%	71	25.0%	-18.8%	4.4%	50.0%
Arkansas	823	4	64.3%	188	100.0%	21.5%	1.8%	75.0%
Kansas	829	4	59.1%	288	75.0%	4.4%	3.0%	50.0%
Mississippi	873	4	58.7%	3,716^{T}	100.0%	32.6%	2.1%	50.0%
Connecticut	883	5	41.2%	104	0.0%	-32.5%	0.7%	40.0%
Iowa	895	4	54.7%	90	75.0%	15.6%	2.5%	50.0%
Oregon	1,000	5	42.6%	257	20.0%	-15.1%	3.4%	40.0%
Oklahoma	1,205	5	68.1%	562	100.0%	13.9%	2.9%	60.0%
Kentucky	1,305	6	64.4%	501	83.3%	3.5%	3.2%	66.7%
South Carolina	1,321	7	56.6%	7,591^{TT}	85.7%	23.0%	1.3%	42.9%
Louisiana	1,384	6	59.8%	4,041	83.3%	14.3%	7.6%	66.7%
Alabama	1,436	7	63.6%	10,146^{TT}	71.4%	-5.3%	3.9%	57.1%
Colorado	1,436	7	45.0%	10,921^{TTT}	42.9%	3.5%	4.5%	42.9%
Maryland	1,474	8	34.4%	1,067	25.0%	7.6%	8.8%	25.0%
Minnesota	1,505	8	47.6%	7,238^{TT}	50.0%	3.7%	1.7%	37.5%
Wisconsin	1,541	8	50.0%	7,610^{TT}	50.0%	1.2%	5.6%	25.0%
Missouri	1,654	8	60.6%	9,583^{TT}	75.0%	1.1%	8.5%	50.0%

Political geography impacts the utilities that the heuristic awards to the players. For example, all five districts in Connecticut (41.2% R voters) are won by D, regardless of who bisects first, whereas all five districts of Oklahoma (68.2% R voters) are awarded to R. These differences are due to the spatial distribution of voters across the states.

The combined effects of the DG setting and the heuristic bisection strategies can be inferred by comparing against the CN setting results. For each instance, the seat-share from the CN setting with the same overall vote-share and number of districts is reported in Tables 4.3 and 4.4 (column 10). Comparing R's seat-share from the DG heuristic (column 6) to R's seat-share from the CN setting, it is clear that the DG setting gives R a greater advantage. When D draws first, R's seat-share in the DG setting is greater than its seat-share in the CN setting in the vast majority of states (13 out of 18). However, the results are mixed when R draws first. Overall, R benefits from the DG setting, especially as the second player. Tables 4.3 and 4.4 also report the total time spent executing Algorithm 4.1 for each instance. States with more census tracts and districts tend to reach the 1 hour time limit more often than those with fewer.

Table 4.4: The bisection protocol (when R draws the first round) with the MIP-based heuristic (Section 4.3.1) is applied to U.S. states with 4 to 8 congressional districts at the census tract-level. Abbreviations include EG for efficiency gap, PA for partisan asymmetry, and CN for the continuous non-geometric setting. *Positive efficiency gap is in favor of R; negative is in favor of D. Superscripts ^{T}, ^{TT}, and ^{TTT} indicate one, two, or three iterations of the MIP solver reached the one-hour time limit.

State	Number of census tracts	Number of districts	Overall R vote-share	Time taken (s)	R seat-share	EG*	PA	CN R seat-share
Utah	716	4	61.3%	72	75.0%	2.1%	11.4%	50.0%
Nevada	778	4	48.7%	51	25.0%	-18.8%	4.3%	50.0%
Arkansas	823	4	64.3%	335	75.0%	-5.0%	6.0%	75.0%
Kansas	829	4	59.1%	76	75.0%	2.5%	5.9%	50.0%
Mississippi	873	4	58.7%	639	50.0%	-19.1%	2.2%	50.0%
Connecticut	883	5	41.2%	87	0.0%	-32.5%	0.7%	40.0%
Iowa	895	4	54.7%	81	50.0%	-9.1%	3.4%	50.0%
Oregon	1,000	5	42.6%	136	20.0%	-15.1%	3.4%	40.0%
Oklahoma	1,205	5	68.1%	350	100.0%	13.9%	2.9%	80.0%
Kentucky	1,305	6	64.4%	333	83.3%	3.4%	3.0%	66.7%
South Carolina	1,321	7	56.6%	460	57.1%	-6.5%	2.3%	57.1%
Louisiana	1,384	6	59.8%	398	83.3%	14.3%	7.5%	66.7%
Alabama	1,436	7	63.6%	$9{,}730^{TT}$	85.7%	8.9%	2.7%	71.4%
Colorado	1,436	7	45.0%	868	42.9%	1.5%	2.9%	57.1%
Maryland	1,474	8	34.4%	1,267	25.0%	7.8%	7.2%	37.5%
Minnesota	1,505	8	47.6%	$3{,}843^{T}$	50.0%	3.9%	2.1%	50.0%
Wisconsin	1,541	8	50.0%	671	75.0%	25.0%	9.7%	75.0%
Missouri	1,654	8	60.6%	$8{,}090^{TT}$	75.0%	1.0%	6.7%	75.0%

4.6 Conclusions

This chapter presents a method to adapt the bisection and ICYF protocols for practical political districting. The heuristics simulate how the two players may engage with limited computational resources. Based on the application of the heuristics for the two protocols, the following key observations are made:

- The case study on congressional districting in Iowa indicates that the proposed heuristics for both the protocols can produce district plans that are fairer than Iowa's 115th congressional districts.
- The bisection heuristic can generate feasible district plans for 18 census tract-level instances. The bisection heuristic tends to give the second player a seat-share advantage in these experiments. Overall, the Republican player tends to benefit from the DG setting and the preference for compact districts due to their spatial voter distribution.

One limitation of this study is that the proposed heuristic strategies for the DG setting may differ from optimal play. The bisection heuristic replicates the vote-share allocation in the CN setting, which may not be the best allocation strategy in the DG setting. Furthermore, while the compactness objective helps alleviate feasibility issues, it may unintentionally bias the district plan in favor of a more spatially distributed party. Future research could improve the computational efficiency of implicit enumeration or develop other techniques to efficiently compute bisection equilibria for large, practical instances.

Chapter 5

Highly Connected Graph Partitioning

5.1 Introduction

Graph partitioning (GP) is the fundamental problem of dividing a graph into smaller parts. GP is important in several application domains such as community detection in social networks and decentralization of power and transportation networks. In some applications, it is desirable that the subgraph induced by each part maintains a minimum level of (vertex) connectivity, Q. Here, a graph's connectivity is measured by the minimum number of vertices whose removal disconnects the graph into 1-connected (or *simply connected*) components. For example, in community detection in social networks, each part (or community) is cohesive if its connectivity is high (Orman and Karadeli, 2015). A cohesive community offers enhanced accessibility through multiple vertex-disjoint paths connecting different portions of the community. Even if $Q - 1$ community members are inactive or not responsive, the rest of the community remains simply connected. In partitioning a power network, requiring minimum connectivity ensures that each part is fault-tolerant to vertex failures, i.e., it requires the failure of at least Q vertices to disconnect a part (Hamad et al., 2011). Hence, high connectivity within graph partitioning is valuable in infusing cohesion and resilience into the parts. In constraint-based modeling, requiring each part to be Q-connected for a pre-determined $Q \geq 1$ provides a lever in choosing the desired amount of cohesion and resilience to infuse within the parts.

Despite its potential benefits, existing research on modeling and solving GP with Q-connectivity constraints is limited. The current body of research mainly focuses on providing theoretical conditions for the existence of a *Q-proper partition* (i.e., a partition of a graph into Q-connected parts), primarily when the input graph is dense (Borozan et al., 2016; Ferrara et al., 2013). For example, Borozan et al., 2016 showed that a graph with n vertices can be partitioned into a bounded (but unspecified) number of 2-connected parts if its minimum vertex degree is at least $\sqrt{n}$. While such findings shed light on the existence of a 2-proper partition for this specific class of dense graphs, these findings do not extend to sparse graphs, i.e., graphs with a minimum vertex degree significantly smaller than $\sqrt{n}$, as is often the case in many real-world graphs such as in power and transportation networks. Even if these results are extended to sparse graphs, the resulting partition may not be suitable for practical scenarios. For example, these works overlook fundamental GP considerations, such as pre-specifying the number of parts, incorporating size balance constraints or optimizing a compactness objective. Hence, a method to produce Q-proper partitions for practical applications must be able to handle any graph input and accommodate fundamental GP considerations.

An integer programming (IP) approach offers the flexibility to accommodate any specific graph instance

while incorporating GP considerations. Existing work in IP approaches for GP has often studied the *political districting problem* (PDP), which partitions a graph into compact, (weighted) size-balanced, and 1-connected parts. While the IP approaches for PDP in existing research have been introduced for political redistricting, some approaches are application-agnostic and can be extended to GP with the aforementioned considerations. Particularly, a popular model for PDP (Hess et al., 1965; Swamy et al., 2023) minimizes a widely-used compactness objective known as the *k-median* or *moment-of-inertia* objective, measured by the sum of weighted squared distances from a *root* within each part; the root is a vertex that minimizes its sum of the weighted squared distances to the vertices within the same part. Optimizing this objective ensures that the vertices within each part are close to the part's "center of mass" (Elsner, 1997). Validi et al., 2022 noted that minimizing this objective encourages connectivity by bringing the vertices closer to their part's root. While ensuring simple connectivity in PDP has historically posed computational challenges, recent advancements by Oehrlein and Haunert, 2017 and Validi et al., 2022 have introduced efficient IP formulations and branch-and-cut methods to alleviate these challenges. However, these formulations and methods cannot currently be used to find Q-proper partitions for $Q \geq 2$.

This paper formalizes the *highly connected graph partitioning* (HCGP) problem, which partitions a graph into compact, balanced and Q-connected parts for any $Q \geq 1$. The IP formulation for HCGP proposed in this study combines elements from IP formulations in two existing studies. This work extends Oehrlein and Haunert, 2017's IP formulation beyond simple connectivity to Q-connectivity for any $Q \geq 1$, thereby enabling fault-tolerant partitions. This formulation also extends Buchanan et al., 2015's formulation for the fault-tolerant dominating set problem, extending their Q-connectivity constraint for a single subgraph to every subgraph within a graph partition. Beyond the IP formulation, this paper offers efficient solution methods to solve HCGP. To obtain an optimal solution for HCGP, we present a branch-and-cut method that solves the presented IP model, similar to a separator-based method in Buchanan et al., 2015. A key challenge is that the number of Q-connectivity constraints in the IP model grows exponentially with the number of vertices. To tackle this computational challenge, the proposed branch-and-cut method adds violated cuts on-the-fly using an efficient integer separation algorithm. To achieve this, we extend the procedure proposed by Validi et al., 2022 for the $Q = 1$ case. We show that the integer separation algorithm in the worst case runs in polynomial-time in terms of the number of parts, vertices and edges, invariant of Q. Furthermore, we introduce valid inequalities based on vertex degrees to strengthen the relaxation lower bounds.

This paper also presents a specialized heuristic, called the *ear construction heuristic*, designed to solve HCGP with $Q = 2$. When prioritizing computational time over optimality, this heuristic is valuable in tackling large instances where the exact method may be computationally intractable, as observed for the $Q = 1$ case (Validi et al., 2022). The ear construction heuristic aims to partition a graph into size balanced and 2-connected parts while locally optimizing compactness. To accomplish this goal, we leverage the property that a 2-connected graph contains an *ear decomposition*, which comprises a cycle and a series of paths constructed sequentially. Using this property, the heuristic first constructs 2-connected subgraphs using multiple ear decompositions, with each subgraph serving as a part in the partition. Since any parts formed from the remaining graph may not be 2-connected, this heuristic employs a repair stage that fixes parts that are not 2-connected by locally reassigning vertices. Finally, two improvement stages improve the partition's size balance and compactness, respectively, using local search.

To evaluate the computational performance of the proposed solution methods, we prepare a curated test bed of 42 graph instances by altering graphs derived from various real-world domains. These domains encompass planar graphs such as transportation networks and census tract adjacency graphs, and non-planar

graphs such as power and social networks. The pre-processing stage alters these graphs to create diverse graphs that are at least 2-connected. Ensuring this minimum level of input graph connectivity encourages the problem to have feasible partitions that are highly connected. Most of these processed graphs are sparse, with minimum degrees between two and four. These minimum degrees are considerably lower than the $O(\sqrt{n})$ threshold required for existing research to guarantee the existence of a feasible Q-proper partition (Borozan et al., 2016; Ferrara et al., 2013). The number of vertices in these processed graphs ranges from 20 to 874. The diversity of this test bed in terms of domain, planarity and size enables us to make meaningful observations about the performance of the solution methods under different circumstances.

The computational analysis in this paper evaluates the proposed solution methods' effectiveness in finding optimal or feasible solutions to HCGP. For $Q = 2$, this analysis delves into the impact of graph size and sparsity on the computational performance of the proposed methods. Furthermore, this analysis quantifies the *costs* of enforcing higher connectivity (for Q up to 4) using the proposed branch-and-cut compared to the baseline case of $Q = 1$ for the same graph. These costs are measured in terms of the compromise in the compactness objective and the additional computational time required to solve the problem with higher connectivity. This analysis offers insights into the trade-offs in implementing the proposed IP formulation and solution methods for HCGP.

The rest of this paper is organized as follows. Section 5.2 presents key literature interlinking graph partitioning, graph connectivity, and existing approaches to related problems. Section 5.3 describes graph preliminaries and formally defines HCGP. Section 5.4 introduces an IP formulation for HCGP and provides a branch-and-cut approach to solve it. Section 5.5 presents the ear construction heuristic for solving HCGP with $Q = 2$. Section 5.6 provides computational results from solving HCGP for a variety of instances. Finally, Section 5.7 offers concluding remarks and potential future research directions. The appendix provides supplementary details on the proofs for theoretical results, pseudocodes for the heuristic, and tabulated data for the computational analysis.

5.2 Literature Review

This section reviews key related literature to better understand models and methods at the intersection of graph partitioning and vertex connectivity. Section 5.2.1 outlines literature on graph partitioning (GP) and the value of high connectivity in GP. Section 5.2.2 details existing approaches to GP with high connectivity. Section 5.2.3 reviews IP approaches for high connectivity constraints and the closely related political districting problem (PDP).

5.2.1 Graph Partitioning and High Connectivity

GP is an extensively studied problem of dividing a graph into smaller parts (Buluç et al., 2016; Bichot and Siarry, 2013). As an optimization problem, GP has a range of variants in terms of objectives and constraints. A common objective is the compactness of the parts, which measures the shape of the parts. While compactness can be measured in several ways, two popular formulations are to minimize the weighted number of edges that cross two parts (Hendrickson and Kolda, 2000), and to minimize the sum of weighted squared distances from each vertex to the "center" of its part (Elsner, 1997). In the latter case, the distances can be measured either by the Euclidean distance or the number of edges in a shortest path between two vertices (Diekmann et al., 2000; Simon, 1991). Furthermore, it is common to ensure that the weighted sizes of the parts are relatively close, which is typically modeled as a constraint, called a *size balance* constraint, and

less frequently as an objective (Buluç et al., 2016). These canonical GP considerations are common in a range of applications such as parallel computing (Hendrickson and Kolda, 2000), districting (Ricca et al., 2013), community detection in social networks (Bedi and Sharma, 2016), and decentralization of power networks (Hamad et al., 2011).

Several application-specific studies have considered GP with high vertex connectivity within each part, particularly in community detection and enhancing resilience. In social networks, partitioning a graph into highly connected communities is valuable in applications such as mass communication or product recommendations (Bedi and Sharma, 2016). The works by Orman and Karadeli, 2015; Orman, 2021 and Timofieiev et al., 2008 have defined a Q-connected subgraph in a social network as a cohesive community of users, where the existence of Q vertex-disjoint paths is desired. Maximizing the internal connectivity within each community can enhance communication and facilitate the flow of information in social networks. In power networks, GP creates "islands" or self-sufficient smaller networks resilient to vertex failures. For example, Hamad et al., 2011 introduced a clustering approach that maximizes the connectivities of the parts while partitioning a power network. Furthermore, partitioning a transportation network is useful for distributed traffic management and computing. In resilience-focused partitioning of a transportation network, existing work addresses the problem of dynamic re-partitioning after vertex disruptions (Ghavidelsyooki et al., 2017). These works tailor resilience considerations to their respective application domains, but do not provide a general solution framework for GP with high connectivity requirements.

5.2.2 Approaches to GP With High Connectivity

Existing research in GP with high connectivity requirements focuses on exploring theoretical conditions for the existence of a highly connected partition. These works particularly focus on highly dense input graphs with a high minimum vertex degree δ. Researchers (Borozan et al., 2016; Ferrara et al., 2013) have explored two key questions: Given a graph G with n vertices and minimum vertex degree δ, (i) what is the minimum value of δ for which a Q-proper partition of the graph G exists, and (ii) can the number of parts in the partition be bounded above? Borozan et al., 2016 showed that if $\delta \geq \sqrt{n}$, there exists a 2-proper partition of G into at most n/δ parts. Furthermore, if $\delta \geq \sqrt{c(Q-1)n}$, where $c = 2123/183$ and $Q \geq 2$, there exists a Q-proper partition of G into at most cn/δ parts. These results provide general existence guarantees for highly dense graphs. However, these existence guarantees do not apply to partitioning problems on sparse graphs, nor can they enforce canonical constraints like guaranteeing a specific number of parts or limiting the size of each part. Hence, an algorithmic approach to find a Q-proper partition in a general setting remains an open problem.

A related algorithmic approach is the Highly Connected Subgraph clustering algorithm. This method partitions a graph while ensuring that each part's *edge connectivity* (i.e., the minimum number of edges whose removal disconnects the graph) is greater than half the number of vertices in that part (Hartuv and Shamir, 2000). This algorithm has been successful in applications involving biological networks, including clustering cDNA fingerprints and grouping protein sequences (Bliznets and Karpov, 2017). Like the previously mentioned approaches for dense graphs, this algorithm does not enforce key GP constraints (e.g., requiring a particular number of parts). Moreover, ensuring high edge connectivity may not translate to ensuring high vertex connectivity. For example, consider a graph constructed as follows: consider two complete graphs, introduce an additional vertex, and add an edge from this vertex to every vertex in both graphs. In this composite graph, the edge connectivity is high, i.e., nearly equal to the number of vertices in either of the complete subgraphs. However, the vertex connectivity is one, where the additional vertex acts as a cut vertex.

Hence, an alternative algorithmic approach is needed to ensure high vertex connectivity in graph partitioning.

5.2.3 Algorithmic Approaches to High Connectivity and GP

Formulating high vertex connectivity constraints in an IP model has predominantly been studied with the goal of finding a highly connected dominating set in wireless networks. In this problem, the goal is to construct a fault-tolerant *backbone* subgraph, wherein the vertices in the backbone serve as intermediary transmission points. Existing research in this area has proposed formulations for constructing a 2-connected backbone (Li et al., 2012; Dai and Wu, 2006), which have subsequently been extended to constructing a Q-connected backbone (Ahn and Park, 2015; Buchanan et al., 2015) for any $Q \geq 2$. These models leverage vertex separator-based constraints to ensure Q-connectivity within the backbone subgraph. However, the challenge faced by HCGP is that it requires all the part subgraphs to be Q-connected, rather than a single subgraph representing the backbone. Existing work has not applied high connectivity constraints to HCGP, and the computational costs of ensuring them remain unclear.

Existing IP models for GP are often tailored to solving PDP, a special case of HCGP when $Q = 1$. Refer to Ricca et al., 2013 for a survey. Hess et al., 1965 proposed a classical model for PDP. Since then, several variants of PDP have emerged, including sales force districting (Salazar-Aguilar et al., 2011) and police districting (D'Amico et al., 2002). With its higher connectivity requirements, HCGP presents a more challenging generalization of PDP.

By examining the exact methods used for solving PDP, we can develop effective methods to address the challenge posed by HCGP. Notably, Oehrlein and Haunert, 2017 and Validi et al., 2022 have introduced state-of-the-art formulations and methods for ensuring 1-connectivity in PDP, a longstanding challenge. Oehrlein and Haunert, 2017 have formulated an IP based on vertex separators, and Validi et al., 2022 have proposed an efficient branch-and-cut algorithm to solve this formulation by strengthening its valid cuts. While these studies address the computational challenge of requiring 1-connectivity in each part, the computational costs of extending these method to Q-connectivity for any $Q \geq 1$ (e.g., by leveraging the Q-connected backbone formulation from Buchanan et al., 2015) have not been studied.

If solution time is a priority over optimality, several heuristics have been proposed to solve PDP. A popular constructive heuristic is the multi-kernel growth method (Vickrey, 1961), which grows the parts by incrementally adding neighboring vertices. To extend this approach to HCGP, it is necessary to ensure that each growth step maintains Q-connectivity within the parts. Furthermore, there are several improvement heuristics for PDP, including local search (King et al., 2012), simulated annealing (D'Amico et al., 2002), tabu search (Bozkaya et al., 2003), and multilevel algorithms (Swamy et al., 2023). These heuristics modify an initial solution (e.g., one obtained with a constructive heuristic) to optimize an objective (e.g., compactness). To adapt these methods to HCGP, designing a heuristic that maintains Q-connectivity after every improvement step is essential.

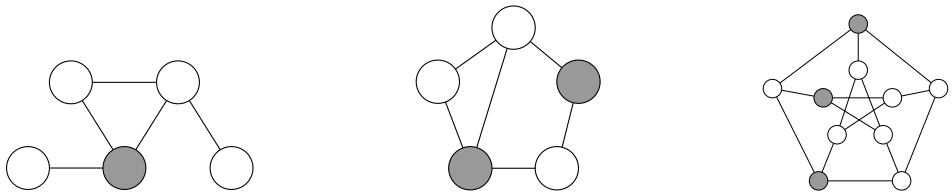
5.3 Problem Description

This section presents a mathematical description of the highly connected graph partitioning (HCGP) problem. Section 5.3.1 defines mathematical terminology and notation related to graphs and connectivity. Section 5.3.2 applies these graph theoretic concepts to define HCGP formally. Section 5.3.3 provides a general integer programming (IP) model for HCGP.

5.3.1 Graph Preliminaries

We introduce the graph terms and notation used in the rest of the paper. An undirected simple graph $G = (V, E)$ comprises a vertex set V and an edge set $E \subseteq V \times V$. Let the number of vertices and edges be $n = |V|$ and $m = |E|$, respectively. For a subset of vertices $S \subseteq V$, let $N(S) \coloneqq \{v \in V \backslash S : \exists\, u \in S, (u, v) \in E\}$ denote the set of its neighbors. The *degree* of a vertex $i \in V$ is its number of neighbors $|N(\{i\})|$. The *minimum* (*average*) degree δ ($\bar{\delta}$) of a graph is the minimum (average) degree among all its vertices. Furthermore, for a subset of vertices $S \subseteq V$, let $G[S]$ denote the subgraph induced by S. Inversely, for a subgraph G' of G, let $V[G']$ and $E[G']$ denote the set of vertices and edges, respectively, in G'. Let $G - S \coloneqq G[V \backslash S]$ denote the subgraph induced by removing S from G.

We now define concepts related to paths. A *path* P is an ordered sequence of distinct vertices $(a_1, a_2, \ldots, a_{|P|})$ such that $|P| \geq 2$ and each vertex is a neighbor of its succeeding vertex, i.e., $a_{i+1} \in N(\{a_i\})$ for all $i = 1, 2, \ldots, |P| - 1$. In a path P, the subset $P \backslash \{a_1, a_{|P|}\}$ constitutes its *internal vertices*. Given two vertices $s, t \in V$, an s,t-*path* is a path originating from s and terminating at t, i.e., $a_1 = s$, $a_{|P|} = t$. Two s,t-paths P and P' are said to be *internally disjoint* if they share no vertices besides s and t, i.e., $P \cap P' = \{s, t\}$. A *cycle* P is a path with at least three vertices and whose end-points are adjacent, i.e., $a_1 \in N(\{a_{|P|}\})$. A *tree* is a graph that does not contain a cycle.



(a) 1-connected (b) 2-connected (c) 3-connected

Figure 5.1: Graph examples illustrating vertex connectivity; shaded vertices constitute a minimum cutset.

We now formalize concepts related to graph connectivity. A graph is said to be *connected* if there is an s,t-path in G between every pair of distinct vertices $s, (\neq) t \in V$; otherwise, the graph is *disconnected*. A *component* of a graph is a maximal connected subgraph. A *forest* is a graph whose components are trees. Given a graph G, a *cutset* (or a *vertex separator*) is a subset of vertices $C \subset V$ whose removal disconnects the remaining graph, i.e., $G - C$ is disconnected. If a cutset comprises a single vertex, that vertex is called a *cut vertex*. A *minimum cutset* is a cutset of the smallest size. Let $\mathcal{C}(G)$ be the collection of all cutsets of G. For a given positive integer $Q \geq 1$, a graph G with n vertices is Q-*connected* if $n \geq Q + 1$ and the size of every cutset of G is at least Q, i.e., $|C| \geq Q$ for all $C \in \mathcal{C}(G)$; otherwise, it is called Q-*disconnected*. Figure 5.1 illustrates examples of Q-connected graphs for $Q \in [3]$. Further, the *connectivity* of graph G, given by $\kappa(G) \coloneqq \arg\min_{C \in \mathcal{C}(G)} |C|$, is the size of its minimum cutset. By convention, a complete graph G is considered $(n-1)$-connected, and $\kappa(G) = n - 1$. Furthermore, a disconnected graph G is 0-connected, and $\kappa(G) = 0$.

5.3.2 Highly Connected Graph Partitioning (HCGP)

The *highly connected graph partitioning* (HCGP) problem divides a graph into a given number of parts, denoted by $K \geq 2$, that are balanced in size, highly connected, and optimally compact. HCGP is formally defined as follows. Consider a graph $G = (V, E)$ with vertex set V and edge set E. Each vertex $i \in V$

has an associated weight $p_i \geq 0$. For a subset of vertices $S \subseteq V$, let $size(S) \coloneqq \sum_{i \in S} p_i$ denote the total weighted size of S. HCGP requires that the weighted size of each part should be within given limits. Let L and U be the lower and upper limits on the weighted sizes, respectively. Further, HCGP requires that the connectivity of each part meets a minimum threshold, denoted $Q \geq 1$ (e.g., when $Q = 1$, each part must be simply connected as in PDP).

A feasible partition to HCGP is now defined. A *K-partition* $\mathcal{P}$ of $G = (V, E)$ is a collection of vertex subsets $V^{(1)}, V^{(2)}, \ldots, V^{(K)}$ that cover G, i.e., $\cup_{k=1}^{K} V^{(k)} = V$, and are pair-wise disjoint, i.e., for $k, k' \in [K]$, $V^{(k)} \cap V^{(k')} \neq \emptyset \Leftrightarrow k = k'$.

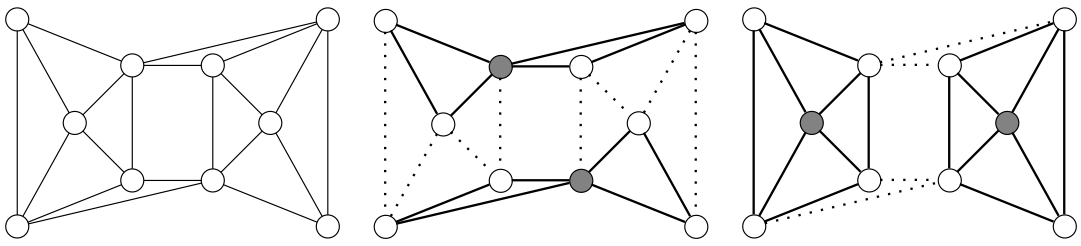
Definition 1. *A K-partition $\mathcal{P}$ is said to be:*

- L,U-balanced *if the weighted size of each part is within the range $[L, U]$, i.e., $size(V^{(k)}) \in [L, U]$ for all $k = 1, 2, \ldots, K$.*
- Q-proper *if the subgraph $G[V^{(k)}]$ is Q-connected for all $k = 1, 2, \ldots, K$.*

HCGP finds an L, U-balanced Q-proper K-partition of G that minimizes a compactness objective defined as follows. For every pair of vertices i and j, let $w_{ij} \geq 0$ denote the distance between them. Given a K-partition $\mathcal{P} = \{V^{(k)}\}_{k \in [K]}$, for each part $k \in [K]$, a *root* $r_k \in V^{(k)}$ is a vertex that minimizes the sum of distances to the rest of the part, i.e., $r_k \coloneqq \arg\min_{i \in V^{(k)}} \sum_{j \in V^{(k)}} w_{ij}$. The *compactness* of a partition is calculated as the sum of distances from the roots to the rest of their respective parts, given as

$$\phi_{comp}(\mathcal{P}) \coloneqq \sum_{k=1}^{K} \sum_{j \in V^{(k)}} w_{r_k j}. \tag{5.1}$$

This objective is commonly referred to as the k-median or p-median objective. As prescribed in PDP models, such as in Hess et al., 1965, we set the pair-wise vertex distance $w_{ij} \coloneqq p_j d_{ij}^2 / size(V)$, where d_{ij} is the number of edges in a shortest path between i and j in G; the denominator $size(V)$ is a normalization term. If the geometric embedding of the vertices is known, it is also common for d_{ij} to measure the Euclidean distance between the coordinates of i and j. However, we consider the shortest path-based distance for an embedding-agnostic graph instance.



(a) Graph G. (b) Optimal solution for $Q = 1$. (c) Optimal solution for $Q = 3$.

Figure 5.2: 1-connected versus 3-connected parts: An example illustrating optimal $5, 5$-balanced Q-proper 2-partitions of a graph G for $Q \in \{1, 3\}$; assume each vertex's weight to be one. Solid lines are internal edges in each part and dotted lines are inter-part edges. Each part's root vertex is shaded.

Figure 5.2 illustrates optimal solutions to HCGP with $Q = 1$ and with $Q = 3$ for a sample graph instance depicted in Figure 5.2a. Compared to the 1-proper partition in Figure 5.2b, the 3-proper partition in Figure 5.2c exhibits more robust parts with multiple vertex-disjoint paths within each part. Hence, ensuring

3-connectivity within the parts is essential to guarantee a robust partition. Further, note that both partitions have the same compactness value of 0.8. Hence, for this specific graph, ensuring 3-connectivity does not inflict any cost in terms of compactness compared to ensuring 1-connectivity.

5.3.3 Modeling HCGP as an Integer Program

We now present an IP formulation for HCGP. HCGP can be modeled as an extension of the connectivity-free PDP model given by Hess et al., 1965, referred to as the *Hess model* by Validi et al., 2022. This model captures the balance constraint and the compactness objective in HCGP, but not the Q-connectivity constraints. The primary decision variables in this IP select K roots and assign each vertex to a root, denoted as

$$x_{ii} = \begin{cases} 1, & \text{if vertex } i \in V \text{ is chosen as a root,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if vertex } j\ (\neq i) \in V \text{ is assigned to root } i \in V, \\ 0, & \text{otherwise.} \end{cases}$$

The Hess model is given as

$$\text{(PDP) Minimize} \sum_{i,j \in V} w_{ij} x_{ij} \tag{5.2a}$$

$$\text{such that } L x_{ii} \leq \sum_{j \in V} p_j x_{ij} \leq U x_{ii} \qquad \forall i \in V, \tag{5.2b}$$

$$\sum_{i \in V} x_{ii} = K, \tag{5.2c}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V, \tag{5.2d}$$

$$x_{ij} \leq x_{ii} \qquad \forall i, j \in V, \tag{5.2e}$$

$$\mathbf{x} \in \{0,1\}^{|V| \times |V|}. \tag{5.2f}$$

The objective in (5.2a) minimizes compactness. Constraints (5.2b) ensure that the weighted size of every part is within the given range $[L, U]$. Constraint (5.2c) requires exactly K vertices to be chosen as roots, and constraints (5.2d) ensure that every vertex is assigned to exactly one root. Constraints (5.2e) ensure that a vertex j can be assigned to a vertex i only if i is chosen as a root. Constraint (5.2f) defines the binary nature of the variables. We denote $\mathcal{P}_{HESS} \coloneqq \{\mathbf{x} \in \{0,1\}^{|V| \times |V|} : \mathbf{x}$ satisfies constraints (5.2b) – (5.2e)$\}$ as the polytope of feasible solutions to the Hess model.

The Hess model finds an optimally compact L, U-balanced K-partition of a graph. HCGP additionally requires the subgraph induced by each part to be Q-connected (for $Q \geq 1$), which is the focus of this paper. Given a solution $\hat{\mathbf{x}} \in \mathcal{P}_{HESS}$, let $R(\hat{\mathbf{x}}) \coloneqq \{i \in V : \hat{x}_{ii} = 1\}$ denote the set of all roots. For each root $r \in R(\hat{\mathbf{x}})$, let $V_r \coloneqq \{i \in V : \hat{x}_{ri} = 1\}$ denote the set of vertices assigned to r. Then, $\hat{\mathbf{x}}$ is *Q-proper* if the subgraph $G[V_r]$ is Q-connected for every root $r \in R(\hat{\mathbf{x}})$. HCGP is generally expressed as

$$\text{(HCGP) Minimize} \sum_{i,j \in V} w_{ij} x_{ij} \tag{5.3a}$$

$$\text{such that } \mathbf{x} \in \mathcal{P}_{HESS}, \tag{5.3b}$$

$$\mathbf{x} \text{ is } Q\text{-proper.} \tag{5.3c}$$

In existing research (Shirabe, 2009; Oehrlein and Haunert, 2017; Validi et al., 2022), formulating constraints (5.3c) as a (mixed) integer program has been limited to the $Q = 1$ case. This paper provides an IP formulation for constraints (5.3c) for any $Q \geq 1$.

5.4 Exact Method

This section presents an IP formulation for HCGP and a branch-and-cut method to solve it efficiently. To enable the IP formulation, Section 5.4.1 presents a characterization of a Q-proper partition (for $Q \geq 1$) based on vertex separators. Using this characterization, Section 5.4.2 presents a quadratic-constrained IP formulation for the Q-connectivity constraints (5.3c). Section 5.4.3 presents valid inequalities based on the property that the degree of a Q-connected subgraph should be at least Q. Finally, Section 5.4.4 presents an integer separation algorithm solved within a branch-and-cut algorithm for HCGP. Appendix D.1 contains proofs for the theoretical results presented in this section.

5.4.1 Characterization of a Q-Proper Partition

In a Q-proper partition, by definition, the size of every vertex separator within each part must be at least Q. This requirement can be expressed by considering pairs of vertices in a graph and verifying this condition for each vertex separator that disconnects the given pair. Given two vertices a and b in a graph G, an *a, b-separator* in G is defined as follows.

Definition 2. *a, b-separator in $G(V, E)$ (Oehrlein and Haunert, 2017). For $a, b \in V$, a subset of vertices $C \subset V \backslash \{a, b\}$ is an a, b-separator in G if there is no a, b-path in $G - C$.*

Let $\mathcal{C}_{a,b,G}$ denote the collection of all a, b-separators in G. Note that $\mathcal{C}_{a,b,G} = \emptyset$ if $a = b$ or $a \in N(\{b\})$. In a complete graph, and only in a complete graph, $\mathcal{C}_{a,b,G} = \emptyset$ for every distinct $a, b \in V$. Furthermore, a separator is *minimal* if it cannot be reduced in size while it remains a separator. Figure 5.3 illustrates a minimal a, b-separator for two vertices a, b in a 5×5 grid graph.

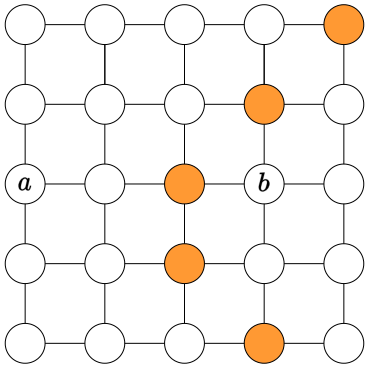


Figure 5.3 A minimal a, b-separator represented by shaded vertices in a 5×5 grid graph.

Oehrlein and Haunert, 2017 gave Definition 2 to formulate 1-connectivity requirements, which we now extend to Q-connectivity. The goal is to characterize each part's Q-connectivity with separators defined on the whole graph. Given the set of all separators for all pairs of vertices in graph G, a Q-proper partition can be characterized as follows. A K-partition is Q-proper if and only if each part $k \in [K]$ has at least $Q + 1$

vertices, and for any pair of vertices a and b within part k, every a, b-separator in G intersects with at least Q vertices within part k.

Theorem 3. *(Characterization of a Q-proper partition) Given a graph G, positive integers $K \geq 2$ and $Q \geq 1$, a K-partition $\mathcal{P} = \{V^{(k)}\}_{k \in [K]}$ is Q-proper if and only if for every $k \in [K]$,*

(i) $|V^{(k)}| \geq Q + 1$, *and*

(ii) for every distinct $a, b \in V^{(k)}$, $|C \cap V^{(k)}| \geq Q$ for every a, b-separator $C \in \mathcal{C}_{a,b,G}$.

Theorem 3 is conceptually similar to Buchanan et al., 2015's result that characterizes a Q-connected d-dominating set for $d \geq Q$; here, a d-dominating set is a subset of vertices S such that every vertex in $G - S$ has d neighbors in S. The similarity is in applying Buchanan et al., 2015's result to each part's vertices (i.e., $S = V^{(k)}$ for $k \in [K]$). However, unlike Buchanan et al., 2015's requirement for a subset to be d-dominating (for $d \geq Q$), Theorem 3 does not impose the dominating constraint on each part's vertices. Furthermore, Buchanan et al., 2015's result is stated for any minimum cutset C, whereas Theorem 3 only considers a subset of cutsets, $\mathcal{C}_{a,b,G}$, for vertex pairs a, b within each part. Hence, for characterizing a Q-proper partition, Theorem 3 differs from Buchanan et al., 2015's consideration of subsets by relaxing the dominating constraint, but restricts the number of cutsets examined. The proof for Theorem 3 is presented in Appendix D.1.1.

5.4.2 Q-CUT Formulation

We express the Q-connectivity constraints (5.3c) as an integer program using the characterization of a Q-proper partition in Theorem 3. Our formulation is a hybrid of the 1-connectivity constraints for graph partitioning from Oehrlein and Haunert, 2017 and the Q-connectivity constraints for the Q-connected d-dominating set problem from Buchanan et al., 2015. Oehrlein and Haunert, 2017's *CUT* formulation is given as

$$\mathbf{x} \in \mathcal{P}_{HESS}, \tag{5.4a}$$

$$\sum_{c \in C} x_{r,c} \geq x_{r,a} \quad \forall\ r \in V,\ a \in V \backslash \{r\},\ C \in \mathcal{C}_{r,a,G}. \tag{5.4b}$$

This formulation defines a constraint for every vertex separator between every vertex and its root. Constraint (5.4a) defines a feasible solution to the Hess model. Constraints (5.4b) ensure that for every pair of vertices $r, a \in V$ and for every r, a-separator C in G, if r is a root and a is assigned to it, then at least one of the vertices in C must also be assigned to r. This constraint ensures that the subgraph induced by each part is connected (i.e., the partition is 1-proper).

Furthermore, Buchanan et al., 2015's formulation creates a Q-connected subgraph as follows. Binary variables select a subset of vertices: for $i \in V$, $z_i = 1$ if i is selected; $z_i = 0$, otherwise. Constraints (10) from Buchanan et al., 2015 are given as

$$\sum_{c \in C} z_c \geq Q z_a\ z_b \quad \forall\ a \in V,\ b \in V \backslash \{a\},\ C \in \mathcal{C}_{a,b,G}. \tag{5.5}$$

For every pair of selected vertices a and b, this constraint ensures that at least Q vertices in every a, b-separator are also selected. Collectively, these constraints select vertices that induce a Q-connected subgraph.

Blending constraints (5.4b) and (5.5), and exploiting the separator-based characterization of a Q-proper

partition in Theorem 3, we create the Q-CUT formulation for HCGP given as

$$\mathbf{x} \in \mathcal{P}_{HESS}, \tag{5.6a}$$

$$\sum_{c \in C} x_{rc} \geq Q x_{ra}\ x_{rb} \quad \forall\ a \in V,\ b \in V \backslash \{a\},\ r \in V,\ C \in \mathcal{C}_{a,b,G}, \tag{5.6b}$$

$$\sum_{i \in V} x_{ri} \geq (Q+1) x_{rr} \quad \forall\ r \in V. \tag{5.6c}$$

This formulation defines a quadratic constraint for every separator between every pair of vertices assigned to the same root. Constraint (5.6a) defines a feasible solution to the Hess model. Constraints (5.6b) ensure that for every trio of vertices $a, b, r \in V$ such that $a \neq b$ and for every a,b-separator C in G, if r is a root and both a and b are assigned to it, then at least Q of the vertices in C must also be assigned to r. Constraints (5.6c) ensure that at least $Q+1$ vertices are assigned to each part. The following corollary shows that constraints (5.6b)-(5.6c) ensure Q-connectivity in each part.

Corollary 1. *Given a graph G and $K \geq 2$, consider a solution to the Hess model $\hat{\boldsymbol{x}} \in \mathcal{P}_{HESS}$. For $Q \geq 1$, $\hat{\boldsymbol{x}}$ is Q-proper if and only if $\hat{\boldsymbol{x}}$ satisfies constraints (5.6b)-(5.6c).*

Constraints (5.6b) are quadratic constraints that can be linearized using McCormick envelopes (McCormick, 1976) by introducing a new variable $y^r_{ab} = x_{ra}\ x_{rb}$ and additional constraints given by

$$y^r_{ab} \leq x_{ra} \quad \forall\ a \in V,\ b \in V \backslash \{a\},\ r \in V, \tag{5.7a}$$

$$y^r_{ab} \leq x_{rb} \quad \forall\ a \in V,\ b \in V \backslash \{a\},\ r \in V, \tag{5.7b}$$

$$y^r_{ab} \geq x_{ra} + x_{rb} - 1 \quad \forall\ a \in V,\ b \in V \backslash \{a\},\ r \in V. \tag{5.7c}$$

Similar to the CUT formulation for $Q = 1$, the Q-CUT formulation for $Q \geq 1$ has an exponential number of constraints due to the number of possible separators. Instead of enumerating all separators, Section 5.4.4 provides a branch-and-cut method that solves the Hess model, and adds violated constraints on-the-fly for solutions that are not Q-proper.

5.4.3 Minimum Degree Valid Inequalities

We now introduce valid inequalities to strengthen the linear programming lower bound of the formulation. We use the following property.

Proposition 2. *For $Q \geq 1$, the minimum vertex degree of a Q-connected graph is at least Q.*

The correctness of this property can be seen by assuming the contrary that the degree of some vertex is less than Q. Then, the neighborhood set of this vertex is a cutset of size less than Q, which contradicts the fact that the given graph is Q-connected. Using this property, the degree-constraints ensure that every vertex has at least Q neighbors in the same part, given as

$$\sum_{j \in N(\{i\})} x_{rj} \geq Q x_{ri} \quad \forall\ i \in V,\ r \in V. \tag{5.8}$$

There are n^2 constraints in (5.8), comparable in size to the Hess model with $n^2 + 2n + 1$ constraints. Furthermore, note that constraints (5.8) ensure that each part contains $Q+1$ vertices (e.g., the root and at

least Q of its neighbors), and therefore implies (5.6c). Hence, the Q-*CUT* formulation can be expressed by the Hess model (5.2a)-(5.2f), and constraints (5.6b) and (5.8).

5.4.4 Separation Algorithm

We propose a branch-and-cut approach to solve HCGP, which addresses the challenges posed by the exponential number and quadratic nature of the separator constraints (5.6b). This approach starts by relaxing the separator constraints and solving the Hess model (5.2a)-(5.2f), and the minimum degree constraints (5.8). By solving the relaxed problem, this approach avoids adding the exponential number of constraints (5.6b). When the branch-and-cut approach encounters an integer feasible solution that is not Q-proper, it solves an integer separation problem. This approach identifies a set of violated separator constraints and adds them on-the-fly. To address the quadratic nature of separator constraints (5.6b), this approach adds linearized versions of these violated constraints; the correctness of the linearization is shown in Theorem 4. This dynamic and linearized addition of constraints effectively handles the computational intractability of the model to find optimal or near-optimal solutions to HCGP.

This section presents an integer separation algorithm for HCGP, extending the approach by Validi et al., 2022 for the $Q = 1$ case to accommodate any $Q \geq 1$. This approach also extends Buchanan et al., 2015's separator-based cutting planes method for the fault-tolerant dominating set problem. Given an integer feasible solution $\hat{\mathbf{x}}$ to the relaxed model, if $\hat{\mathbf{x}}$ is not Q-proper, this algorithm identifies a subset of separator constraints (5.6b) that $\hat{\mathbf{x}}$ violates. Each such violated constraint is identified by selecting the corresponding vertices a, b and r (where $a \neq b$), and an a, b-separator C in G. To find a minimal a, b-separator in G, we use the procedure outlined in Fischetti et al., 2017, initially proposed by Validi et al., 2022 for the $Q = 1$ case. We demonstrate how this procedure extends to any $Q \geq 1$.

The computational efficiency of the proposed integer separation algorithm stems from examining the structure of a Q-disconnected part. When a part with a root $r \in R(\hat{\mathbf{x}})$ is connected but not Q-connected, it contains a minimum cutset D such that $|D| < Q$. Removing D from that part disconnects it into more than one component. A naive implementation of the algorithm proceeds as follows: from every pair of such components, select arbitrary vertices a and b. Then, identify a minimal a, b-separator in G, and add the corresponding violated constraint. However, this approach can potentially iterate through $O(n^2)$ pairs of components in the worst case. Alternatively, assuming that the root is not a part of the cutset (i.e., $r \notin D$), an alternative approach is to set $a = r$. Then, from every component that does not contain r, an arbitrary vertex b is selected. A minimal r, b-separator in G is identified, and the corresponding violated constraint is added. This alternative approach would add $O(n)$ violated constraints in the worst case. Hence, to enhance computational efficiency, the separation algorithm first checks whether $r \notin D$, in which case that part is called *root-resilient*. If the part is root-resilient, it adds the $O(n)$ constraints; otherwise, it adds the pair-wise set of $O(n^2)$ constraints. If the part with root r is not connected, the same procedure is applied to the component of the part that contains r rather than the entire part (i.e., if this component is not Q-connected, then the integer separation algorithm finds a minimum cutset D of the component, classifies the part as root-resilient if $r \notin D$, and adds the corresponding set of violated constraints for that component); to promote Q-connectivity for the entire part, the algorithm also adds violated constraints between the component containing r and the other components. This approach diverges from Buchanan et al., 2015's approach by employing Fischetti et al., 2017's separator construction and by incorporating root-resilience.

Algorithm 5.1 presents a pseudocode for the separation algorithm for a given integer solution to the relaxed model $\hat{\mathbf{x}}$. Recall that $R(\hat{\mathbf{x}})$ denotes the set of roots and V_r denotes the set of vertices assigned to a

Algorithm 5.1: Integer Q-Connectivity Separation (G, $\hat{\mathbf{x}}$, Q)

1 **for** $r \in R(\hat{\boldsymbol{x}})$ **do**
2 **for** *every component* G_0 *of* $G[V_r]$ *not containing* r **do**
  `/* the part is disconnected */`
3 let b be an arbitrary vertex of G_0
4 $C \leftarrow$ Minimal separator(G, $V[G_0]$, r, b)
5 add constraint $Q\hat{x}_{rb} \leq \sum_{c \in C} \hat{x}_{rc}$ to the model
6 let G_r be the component of $G[V_r]$ that contains r
7 let D be a minimum cutset in G_r
8 **if** $|D| < Q$ **then**
9 **if** $r \notin D$ **then**
  `/* the part is Q-disconnected and root-resilient */`
10 **for** *every component* G_0 *of* $G_r - D$ *that does not contain* r **do**
11 let b be an arbitrary vertex of G_0
12 $C \leftarrow$ Minimal separator(G, $V[G_0]$, r, b)
13 add constraint $Q\hat{x}_{rb} \leq \sum_{c \in C} \hat{x}_{rc}$ to the model
14 **else**
  `/* the part is Q-disconnected and not root-resilient */`
15 **for** *every pair of components* G_1 *and* G_2 *of* $G_r - D$ **do**
16 let a and b be arbitrary vertices in G_1 and G_2, respectively
17 $C \leftarrow$ Minimal separator($G, V[G_2], a, b$)
18 add constraint $Q(\hat{x}_{ra} + \hat{x}_{rb} - 1) \leq \sum_{c \in C} \hat{x}_{rc}$ to the model

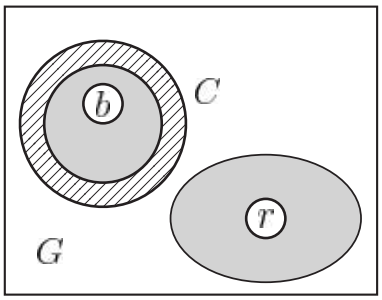
19 **Procedure** *Minimal separator (G, S, a, b) (Fischetti et al., 2017)*
  `/* here, S is a subset of vertices of G containing b, and a ∉ S */`
20 define the neighbors of S to be $\mathcal{N} = N(S)$
21 delete all edges in $E[S \cup N(S)]$ from G
22 find the set $\mathcal{R}$ of vertices that can be reached from a in G
23 **return** $\mathcal{N} \cap \mathcal{R}$

root $r \in R(\hat{\mathbf{x}})$. The algorithm iterates (in line 1) over each part with a root r. First, the algorithm checks whether the part is disconnected. If it is, lines 2-5 add violated constraints for each r, b-separator, where b is an arbitrary vertex selected from each component of the part subgraph that does not contain r. Next, lines 6 and 7 find a minimum cutset D in the component G_r of the part containing r. Note that $G_r = G[V_r]$ if the part is connected. In the special case that G_r is a complete graph, D returns the vertices in G_r without r, rather than a traditionally-defined cutset of G_r. Line 8 verifies whether the size of D is less than Q. If it is, indicating that the part is not Q-connected, lines 9-18 add violated constraints. If the part is root-resilient, lines 10-13 add a constraint between r and an arbitrary vertex b in every component of $G_r - D$. Otherwise, lines 15-18 add a constraint between every pair of vertices a and b from different components of $G_r - D$.

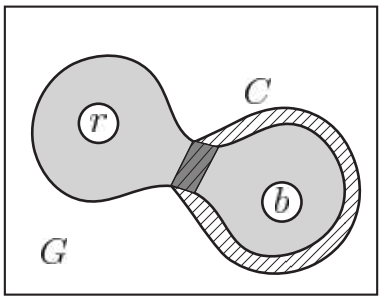
Figure 5.4 depicts an abstract representation of the three cases when Algorithm 5.1 adds violating constraints. The correctness of Algorithm 5.1 is shown in Theorem 4. Let $\mathcal{P}'_{HESS}$ denote the polytope of feasible solutions to the relaxed model given by the Hess model (5.2a)-(5.2f) and constraints (5.8). Note that in Theorem 4, a violated separator constraint is a subset of separator constraints (5.6b) and (5.7a)-(5.7c) violated by a solution to the relaxed model.

Theorem 4. *Given a graph $G = (V, E)$, an integer $Q \geq 1$, and a solution $\hat{\boldsymbol{x}} \in \mathcal{P}'_{HESS}$, Algorithm 5.1 adds violated separator constraints if and only if $\hat{\boldsymbol{x}}$ is not Q-proper.*

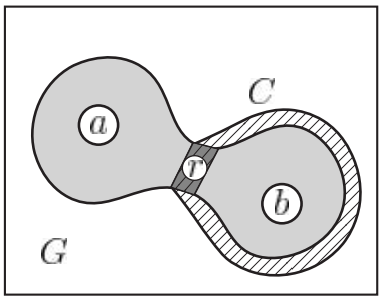
To analyze the run-time of Algorithm 5.1, we consider two cases. First, if all the parts are root-resilient to



(a) The part is disconnected; the shaded regions compose the part with root r and the hatched region is C, a minimal r, b-separator in G.



(b) The part is root-resilient; the shaded region (light and dark) is its component G_r containing r, the dark region is a minimum cutset D, and the hatched region is C, a minimal r, b-separator in G.



(c) The part is not root-resilient; the shaded region (light and dark) is its component G_r containing r, the dark region is a minimum cutset D, and the hatched region is C, a minimal a, b-separator in G.

Figure 5.4: Abstract representations of a Q-disconnected part with root r in graph G considering three cases corresponding to the three sets of constraints added in Algorithm 5.1.

the minimum cutsets found in line 8, adding violated constraints is less time-intensive as it only executes the first 13 lines of Algorithm 5.1. Second, if some parts are not root-resilient, lines 15-18 are executed, which increases the solution time. The following theorem provides the worst case run-time for both cases.

Theorem 5. *Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, integers $Q \geq 1$ and $K \geq 2$, and a solution $\hat{\boldsymbol{x}} \in \mathcal{P}'_{HESS}$, Algorithm 5.1 takes $O(Kmn^{5/3})$ time if all the parts of $\hat{\boldsymbol{x}}$ are root-resilient to the cutsets in line 8 or $K \geq \sqrt[3]{n}$; otherwise, it takes $O(mn^2)$ time.*

While the worst case run time of the separation algorithm is independent of Q, the if condition in line 8 may return True for more solutions for larger Q values. As a result, lines 9-18 would generate and add more violated cuts, leading to a longer overall run time. Furthermore, in the special case where G is planar (i.e., $m = O(n)$), Algorithm 5.1 takes $O(Kn^{8/3})$ time if all the parts in $\hat{\mathbf{x}}$ are root-resilient or $K \geq \sqrt[3]{n}$; otherwise, it takes $O(n^3)$ time.

Even though the theoretical worst case run-time of a single run of the separation algorithm is a large polynomial of m and n, empirical results suggest that the algorithm performs well in practice. Specifically, on average over all the instances solved in Section 5.6, the total run-time of all the runs of the separation algorithm within a single execution of the branch-and-cut only accounts for approximately 14.9% of the branch-and-cut time. Hence, the separation algorithm efficiently enforces Q-connectivity for practical instances of HCGP.

5.5 An Ear-Construction Heuristic for HCGP with $Q = 2$

For large instances of HCGP, a heuristic approach can be more effective if the goal is to find a feasible solution in a shorter time than solving the branch-and-cut method. This section presents a heuristic approach to solve HCGP with $Q = 2$. Motivated by heuristic approaches to the political districting problem (PDP), this heuristic first constructs and then improves a solution to HCGP. The inputs to the heuristic are graph G, the number of parts K, and the lower and upper bounds on the part sizes (L and U, respectively). This heuristic has four stages. The first three stages generate a feasible solution, i.e., a 2-proper L, U-balanced K-partition of G, and the fourth stage improves the compactness objective using local search. The stages are as follows.

(i) The *construct* stage generates an initial partition of G into K connected parts. First, it iteratively creates 2-connected parts using a procedure called *ear construction*, and the rest of the graph comprises a set of connected parts. If the number of parts exceeds K, it iteratively merges neighboring parts until there are K parts; if the number of parts is smaller than K, it restarts the construct stage. If some parts are 2-disconnected, the algorithm proceeds to the *repair* stage; otherwise, the algorithm proceeds to the third stage.

(ii) The *repair* stage fixes the 2-connectivity of each 2-disconnected part by reassigning certain vertices between neighboring parts. These vertices are chosen with the goal of reducing the number of cut vertices within a part. After the repair stage, if all the parts are 2-connected, the heuristic proceeds to the size balance improvement stage; otherwise, the algorithm restarts from the first stage.

(iii) The *size balance improvement* stage uses local search (using vertex reassignments) to improve a size balance objective until all the part sizes are within $[L, U]$. If local search terminates before achieving this size balance (i.e., there are no reassignments that improve size balance while also retaining each part's 2-connectivity), the algorithm restarts from the first stage.

(iv) The *compactness improvement* stage uses local search to optimize the compactness objective, producing a locally optimal 2-proper and size-balanced partition.

Each stage incorporates several random steps. For example, the construct stage initially constructs a set of random 2-connected subgraphs of G. If a stage does not produce a partition that satisfies its designed criteria, the heuristic restarts from the first stage. Furthermore, ordering the first three stages ensures connectivity, 2-connectivity, and size balance in that order. Preliminary computational investigations revealed that this ordering produced a feasible solution faster than other permutations, such as ensuring size balance before 2-connectivity.

The rest of this section is organized as follows. Section 5.5.1 describes the ear construction procedure to construct a 2-connected subgraph from a given graph. Sections 5.5.2 and 5.5.3 describe the construct and repair stages, respectively. Section 5.5.4 defines a size balance objective and outlines the local search-based improvement stages. The pseudocodes for the overall heuristic and the individual stages are given in Appendix D.2.

5.5.1 Ear Construction

We now present a procedure to grow a 2-connected subgraph from a given graph if one exists. Such a 2-connected subgraph composes a 2-connected part in the initial partition assembled in the construct stage described in the next section. To construct a 2-connected subgraph, we utilize a graph structure known as an *open ear decomposition*, defined as follows. An *ear* is an ordered sequence of vertices that have the properties of a path with the only exception that the end-points may coincide. An *open ear* is an ordered sequence of distinct vertices identical to a path. An *open ear decomposition*, illustrated in Figure 5.5, is constructively defined as follows.

Definition 3. *Given a graph H, an open ear decomposition of H is a sequence of ears $P_0, P_1, \ldots, P_s$ (for some integer $s \geq 0$) in H such that*

1. *P_0 is a cycle in H, and*
2. *if $s \geq 1$, for $i = 1, 2, \ldots, s$, P_i is an open ear in H with end-points in $H[P_0 \cup P_1 \cup \cdots \cup P_{i-1}]$.*

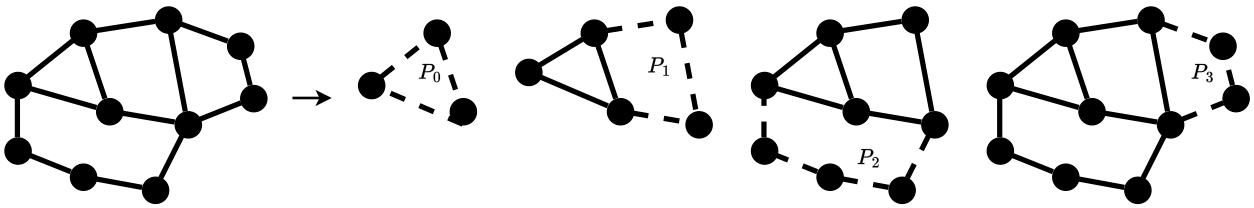


Figure 5.5 An open ear decomposition of a 2-connected graph with 10 vertices into four ears.

The constructive procedure in Definition 3 can be used to create a 2-connected part subgraph in the initial partition of the graph. To achieve this, we use the classical result from Whitney, 1931, which states that a graph with at least two edges is 2-connected if and only if it has an open ear decomposition. In our procedure, rather than decomposing a given graph, we employ the constructive procedure in Definition 3 to generate a sequence of ears to constitute the part subgraph. From Whitney, 1931's result, the vertices traversed by these ears induce a 2-connected subgraph.

To create an initial part, beyond 2-connectivity, it is also desirable to ensure that the part size is at least L. In our procedure, ears are added one at a time. To ensure the minimum size requirement, this iterative addition of ears terminates either when the traversed set of vertices is large enough, i.e., its size is at least L, enough to form a balanced part, or is maximal, i.e., no additional ears can be added. Furthermore, each ear is constructed to increase the part size incrementally to avoid excessively increasing the part size within a single iteration. This goal is achieved by finding a vertex-weighted shortest path (weighted by the vertex sizes) between two randomly chosen vertices that already appearing in the part.

5.5.2 Construct an Initial Partition Using Ear Construction

The construct stage creates an initial K-partition of G. This procedure is analogous to the multi-kernel growth method to find a feasible 1-proper partition for PDP (Vickrey, 1961). This procedure first constructs a series of 2-connected parts by iteratively employing the ear construction procedure in Section 5.5.1, removing that 2-connected subgraph, and proceeding with the remaining graph. This iterative procedure continues until either the remaining graph is empty or is a forest, i.e., there are no more 2-connected subgraphs. Let K' denote the total number of 2-connected subgraphs and trees in the forest. This stage terminates if $K' = K$, indicating that we already have K parts; in the ideal case, all K parts are 2-connected. If $K' \neq K$, there are two possible cases:

- $K' < K$, indicating that there are fewer parts than desired, in which case the construct stage is restarted, or
- $K' > K$, indicating an excess of parts, in which case the procedure iteratively merges a part with a neighboring part until the desired number of K parts are obtained. Each iteration merges the pair of parts with the smallest total size, thereby prioritizing smaller parts to be merged first.

This stage returns a partition with K connected parts, some of which may be 2-disconnected.

5.5.3 Repair 2-Connectivity by Reassigning Small Components

If the partition found in the first stage is not 2-proper, the repair stage attempts to modify the partition to achieve a 2-proper partition. Our preliminary empirical investigations revealed a pattern in a typical

2-disconnected part found in the first stage: removing a cut vertex c of the part would typically result in one large component and one or more smaller components. The repair stage reassigns these smaller components to neighboring parts while leaving c in its original part, at the end of which c is no longer a cut vertex of that part. Figure 5.6 depicts an abstract representation of this reassignment for a given cut vertex c. After repeating this procedure for all the cut vertices of a part, that part becomes 2-connected.

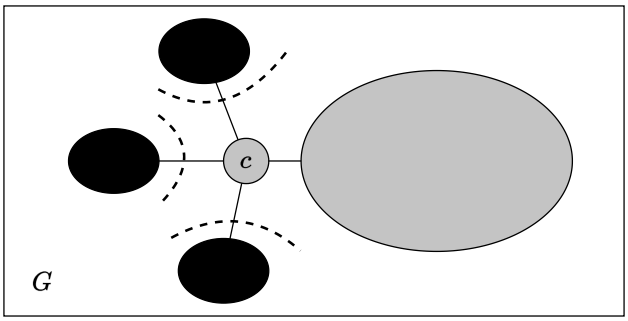


Figure 5.6: Repairing 2-disconnectivity: An abstract representation of a 2-disconnected part (shaded region) with a cut vertex c. The hypothetical removal of c from the part disconnects the part into one large component (lightly shaded region) and several smaller components (dark region) which are reassigned to neighboring parts.

There are two potential issues with this reassignment procedure. First is the possibility that a small component S may not have a neighboring part, in which case the vertices in S cannot be reassigned. Note that this scenario arises only when the given graph G is 1-connected, i.e., the cut vertex c of a 2-disconnected part is also a cut vertex of G. Second, in some cases, the reassignment procedure may introduce cut vertices to the part that receives the vertices. Hence, cycling may occur, where the same subset of vertices is repeatedly reassigned. Here, cycling is heuristically assessed by checking whether the previous 100 iterations are exclusively composed of repeated reassignments. If either of these issues is detected (i.e., a smaller component S does not have a neighboring part or if the cycling condition is true), the procedure terminates and restarts the construct stage. Otherwise, this procedure continues until all the parts become 2-connected.

5.5.4 Improvement Stages: Local Search to Optimize Size Balance and Compactness

Once the heuristic obtains a 2-proper K-partition, it executes two improvement stages. First, starting from a 2-proper partition, this procedure optimizes a size balance objective until all the part sizes are within $[L, U]$. Second, this procedure locally optimizes compactness by minimizing the objective ϕ_{comp} given in Equation (5.1). For a given partition $\mathcal{P} = \{V^{(k)}\}_{k\in[K]}$, the size balance objective is

$$\phi_{bal}(\mathcal{P}) := \max\{\max_{k\in[K]}\{0, size(V^{(k)}) - U\}, \max_{k\in[K]}\{0, L - size(V^{(k)})\}\}. \tag{5.9}$$

Since $L, U \geq 0$, observe that ϕ_{bal} is zero if and only if all the part sizes are within $[L, U]$; otherwise, ϕ_{bal} measures the extent of size imbalance. Therefore, to create an L, U-balanced partition, the local search procedure minimizes ϕ_{bal} until it reaches zero.

The local search procedure uses the popular *Flip* method for PDP (Ricca and Simeone, 2008). This procedure makes small adjustments to a partition by iteratively reassigning vertices, one at a time, from their

current part to a neighboring part. Each potential reassignment is *feasible* if the current and receiving parts maintain their 2-connectivity after the reassignment. Here, 2-connectivity is assessed using Algorithm 11 in Esfahanian, 2013. A feasible reassignment is accepted if it improves the given objective. At the end of the two improvement stages, the heuristic returns an L, U-balanced 2-proper K-partition of G with locally optimal compactness.

5.6 Computational Results

This section presents a computational investigation of the branch-and-cut and heuristic methods for HCGP on various problem instances. We consider 42 graph instances, three values for the number of parts $K \in \{2, 3, 4\}$, two size balance settings, and four values for the minimum connectivity requirement $Q \in [4]$. First, for $Q = 2$, we analyze the effectiveness of both methods in achieving optimal or feasible solutions to HCGP within a limited time. This analysis offers insights into the strengths and weaknesses of each algorithm by assessing their performance with different classes of instances, considering factors such as size and density. Next, using the branch-and-cut method, we empirically measure the cost of ensuring high connectivity for $Q \in \{2, 3, 4\}$. To achieve this goal, we compare the optimal compactness values and the computational time required to optimally solve instances with $Q \in \{2, 3, 4\}$ against instances with $Q = 1$. Taken together, these two analyses provide an empirical understanding of the effectiveness of the methods in achieving optimal or near-optimal solutions to HCGP.

All the computations in this work were performed using a Macbook with 16GB Memory and an M1 chip. The algorithms were written in Python 3.7 and the optimization formulations were solved using Gurobi 10.0. The separation algorithm was implemented within Gurobi's callback functionality. When comparing the computational performance of two algorithms across a set of instances, we take the geometric mean of the ratios of the computational times across the instances, as prescribed by Koch et al., 2022.

The rest of this section is organized as follows. Section 5.6.1 details the datasets and the process used to create instances for HCGP. To determine an efficient implementation strategy for the branch-and-cut method used in this section, Section 5.6.2 presents a preliminary investigation on the number and type of violated constraints added in the separation algorithm. Section 5.6.3 investigates the performance of the exact and heuristic methods for HCGP with $Q = 2$. Section 5.6.4 expands the investigations of the branch-and-cut method for Q values of three and four. Drawing on the results from the previous section, Section 5.6.5 empirically quantifies the cost of ensuring high connectivity (when using the branch-and-cut method) compared to a baseline of ensuring 1-connectivity. Section 5.6.6 illustrates visual examples of highly connected partitions of two graphs.

5.6.1 Datasets and Instance Preparation

This section creates a test bed of instances for HCGP derived from real-world graphs in several domains. First, we describe the 42 graphs compiled from various data sources. Next, to encourage each graph instance to contain feasible highly connected partitions, a pre-precessing stage increases the connectivities of these graph instances. We now detail the graphs selected from the various data sources and the pre-processing stage.

Raw Datasets

We use real-world graphs spanning several domains: transportation networks, census tract adjacency graphs, power networks, social networks, interaction networks, animal social networks, and other miscellaneous graphs. These graphs are obtained from a variety of publicly available data repositories. We select graph instances with fewer than $1,000$ vertices for computational tractability. We now describe each data source by domain.

A transportation network models road intersections as vertices and road links as edges. *Transportation Networks* (Stabler et al., 2020) is a repository of graphs containing urban road networks from cities worldwide, such as Anaheim and Barcelona. We use nine graphs from this repository for our test bed.

A census tract adjacency graph models census tracts in the U.S. as vertices, and edges represent whether two census tracts share a boundary of non-zero length. Each graph corresponds to the census tract graph within a U.S. state. These graphs are used in various applications, most notably in political redistricting. Beyond the adjacency information, U.S. Census Bureau, 2020 also provided the population in each census tract. Among the 50 states, we consider 12 states that each (a) have fewer than $1,000$ census tracts, and (b) were apportioned at least two congressional districts based on the 2020 census.

A power network represents an electricity transmission system comprising buses and generators as vertices, and edges representing transmission lines. We use the *IEEE 300-Bus* network with 247 vertices from ICSEG, 2013 and a power network with 494 vertices from *Network Repository* (Rossi and Ahmed, 2015), a repository with graphs from more than 30 domains.

Social networks are modeled using graphs where vertices represent anonymized social networking platform users, and edges represent pairs of connected users. The *Stanford Network Analysis Project* (SNAP) dataset (Leskovec and Krevl, 2014) contains Facebook networks created by Leskovec and Mcauley, 2012, and we use six graphs from this dataset that have fewer than $1,000$ vertices.

Further, we use interaction and animal social networks from the *Network Repository* (Rossi and Ahmed, 2015). Interactive networks model people as vertices and edges represent physical or virtual interactions. We select five such graphs, including a patient proximity network in a hospital ward and an email communication network. In animal social networks, vertices represent animals and edges represent whether two animals have been in physical proximity. We select three graphs representing an ant colony, wild birds, and dolphins.

Finally, we use five miscellaneous graphs from the *SuiteSparse Matrix Collection* (Davis and Hu, 2011), formerly known as the *University of Florida Sparse Matrix Collection*. Three graphs are Mycielskian graphs with the special properties of being triangle-free and having a high chromatic number. Mycielskian graphs have been used as test instances for problems requiring high connectivity (Buchanan et al., 2015). The other two graphs are highly connected graphs with connectivities of 19 and 81.

Processing Instances

The pre-processing stage aims to increase the connectivities of the input graphs derived from these data sources. We conjecture that highly connected instances are more likely to contain highly connected partitions. Hence, to prepare a test bed of instances for HCGP likely to contain feasible highly connected partitions, we modify each graph instance using vertex deletion or edge addition operations. Note that in the graph creation, at most one edge is allowed to exist between any two vertices, even if the problem domain typically permits the presence of multiple edges, e.g., multiple boundary segments for a pair of census tracts.

We introduce two different pre-processing methods based on the graph's planarity. The instances from transportation networks and census tract adjacency graphs are planar, while those from the rest of the

domains are non-planar. For planar instances, the goal is to produce a 2-connected graph instance while retaining its planarity. To achieve this goal, we select the largest 2-connected component of each graph, and delete the rest of the vertices from the graph. Furthermore, for the non-planar graph instances, the goal is to achieve 4-connected versions, except for power networks where we achieve 2-connected versions due to their sparsity; increasing the connectivity beyond two would significantly alter the structure of a power network. To achieve this goal, we alter a given graph into a Q-connected graph for $Q \in \{2, 4\}$ using the following iterative procedure. In each iteration, we identify the cutsets of sizes less than Q, and the components of the graph that result from the removal of each cutset. Then, we arbitrarily select two vertices (namely the first two vertices in the arrangement of vertex labels) across two different components, and add an edge between them. After adding edges across all pairs of components created by each cutset, and iterating this procedure until there are no cutsets of size less than Q, this graph becomes Q-connected. Note that this procedure does not alter the miscellaneous graphs since their connectivities are above five.

Table 5.1: An overview of 42 processed graph instances in each domain with the ranges in the number of vertices, minimum and average vertex degrees, and the graph connectivities.

Domain	Number of graphs (G)	Size (n)	Minimum degree (δ)	Average degree ($\overline{\delta}$)	Connectivity ($\kappa(G)$)
Transportation	9	20 - 874	2 - 3	3.4 - 4.1	2
Census Tract Adjacency	12	222 - 833	2 - 3	5.2 - 5.6	2
Power Networks	2	247 - 494	2	3.1 - 3.3	2
Social Networks	6	52 - 534	4	7.7 - 22.9	4
Interaction Networks	5	75 - 410	4 - 6	13.9 - 39.9	4 - 6
Animal Social Networks	3	164 - 291	4 - 41	22.4 - 130.9	4 - 41
Miscellaneous	5	95 - 441	6 - 81	15.9 - 81.5	6 - 81

Table 5.1 summarizes the processed graph instances. It reports the graph instances' domain-wise size, degree and connectivity ranges. Table D.1 in Appendix D.3.1 reports the number of vertices removed or the edges added while pre-processing each graph instance. In the rest of this section, the transportation, census tract adjacency and power networks are referred to as *sparse* graphs due to their relatively low minimum degrees (i.e., less than four), and the rest are *dense* graphs. Further, we designate graphs with more than 500 vertices as *large*, while the rest are *small*. Among the 42 graphs, 23 are sparse and 11 are large.

Each graph instance includes vertex weights necessary to incorporate the size-balance constraints. For census tract adjacency graphs, we treat the population residing in each census tract as its vertex weight. For the rest of the graphs, we set the vertex weights to be one. Furthermore, the analysis in this section considers two settings for size balance: one without the size balance constraints, and one requiring that the part sizes do not deviate more than 10% from the average part size, denoted by $\overline{p} := size(V)/K$. This section represents these two settings by a size balance deviation threshold: $\tau = \infty$ without size balance constraints, and $\tau = 0.1$ when requiring $L = 0.9\ \overline{p}$ and $U = 1.1\ \overline{p}$.

5.6.2 Preliminary Analysis of Separation Algorithm

To solve an instance of HCGP, the branch-and-cut method adds on-the-fly constraints generated by the separation algorithm outlined in Algorithm 5.1. In this section, we provide a preliminary analysis of the number and choice of violated constraints added in each run of the separation algorithm. The goal is to determine the settings that result in the fastest computational time. First, we investigate whether adding all the violated constraints generated by Algorithm 5.1 is superior to adding just one violated constraint. Next,

recall that Algorithm 5.1 assesses whether each part is root-resilient; if it is, the algorithm adds $O(n)$ violated constraints in lines 10-13 for root-resilient parts, while it adds $O(n^2)$ violated constraints in lines 15-18 for parts that are not root-resilient. If the root-resilience consideration was not implemented, the default method would add $O(n^2)$ constraints for every integer feasible solution. We conduct comparative experiments by running the algorithm with and without root-resilience considerations to verify its computational benefit.

Table 5.2: Computational times for solving HCGP with seven graph instances, $K = 2$, $Q = 2$ and $\tau = \infty$; the total time spent in callbacks is in parenthesis. Each run of the separation algorithm is executed *with* versus *without* root-resilience considerations, and with adding *all* versus *one* violated constraint. *No feasible solution was found within the time limit.

Graph (G)	n	Time taken with root-resilence (s)		Time taken without root-resilence (s)	
		all	one	all	one
Berlin-Tiergarten	280	32.7 (8.1)	93.0 (7.6)	99.9 (65.8)	151.9 (4.0)
New Mexico	576	80.9 (66.8)	279.0 (9.2)	179.3 (69.5)	256.6 (5.0)
IEEE300Bus	247	110.1 (14.4)	408.9 (8.7)	353.8 (239.6)	549.6 (7.8)
414	150	6.5 (3.2)	10.6 (1.7)	12.1 (6.9)	11.2 (2.7)
infect-hyper	113	2.0 (1.8)	2.8 (0.8)	4.8 (0.8)	4.4 (2.3)
aves-wildbird-network	202	19.5 (8.7)	45.5 (26.4)	39.43 (38.5)	37.8 (22.2)
mycielskian9	383	611.9 (20.2)	1,389.0 (24.9)	3,600* (3,512.2)	1,944.3 (36.8)

For this analysis, we select seven graph instances representing the median number of vertices in each of the seven domains. We solve HCGP with $K = 2$, $Q = 2$, and without the balance constraints. Table 5.2 presents the computational times for the four algorithm settings: *with* versus *without* root-resilience, and including *all* versus *one* violated constraint in each execution of Algorithm 5.1. Each run of the branch-and-cut method was given a time limit of one hour.

Among these four settings, we observe that incorporating root-resilience and adding all the violated constraints produces the fastest computational time in all seven instances. Therefore, this setting is used for the remainder of this section. Furthermore, we make three observations. First, when solving without root-resilience, adding all constraints is more efficient than adding one constraint in the minority (3 out of 7) of instances; the most significant difference is with the mycielskian9 instance. In contrast, when solving with root-resilience, in all seven instances, adding all constraints is more efficient than adding one constraint. This observation suggests that adding all the $O(n^2)$ constraints is computationally expensive without root-resilience, as observed in the callback time. Hence, we recommend adding one constraint at a time when solving without root-resilience. Second, as expected, incorporating root-resilience yields a faster computational time in most (i.e., 12 out of 14) instances. Third, as expected, in most (i.e., 11 out of 14) instances, the algorithm spends less time within the callbacks when one constraint is added than when all constraints are added. However, despite reducing the time spent in callbacks, the branch-and-cut method's overall computational time increases. A possible explanation is that adding all constraints improves the best lower bound, which could enable the algorithm to prune more sub-problems in the branch-and-cut tree, contributing to a faster overall run-time.

5.6.3 Results for HCGP with $Q = 2$

In this section, we investigate the computational performance of the branch-and-cut and heuristic methods for HCGP with $Q = 2$. We solved this problem for the 42 graph instances, three values of the number of parts

$K \in \{2, 3, 4\}$, and two balance settings, $\tau = \infty$ and $\tau = 0.1$. Overall, these combinations of graphs, number of parts and balance settings produce 252 instances. Both methods were given a time limit of one hour.

Table 5.3: Distribution of instances based on solution outcomes for the two methods: optimal, feasible (and suboptimal), infeasible, or inconclusive, categorized by instance type.

Solution outcome	All instances (252)		Sparse instances (138)		Large instances (66)	
	Exact	Heuristic	Exact	Heuristic	Exact	Heuristic
Optimal	204	9	102	0	42	0
Feasible	13	219	2	118	2	65
Infeasible	9	0	9	0	0	0
Inconclusive	26	24	25	20	22	1

Table 5.3 summarizes the performance of both methods by listing the number of instances with outcomes in each of four categories: optimal, feasible (but suboptimal), infeasible, and inconclusive. An instance is *inconclusive* if a method terminates within the time limit with neither a feasible solution nor an infeasibility certificate. Note that for an infeasible instance, the heuristic cannot produce an infeasibility certificate by design and would instead time-out without a feasible solution. Tables D.2 and D.3 in Appendix D.3.2 present more details on these results. Overall, as expected, the heuristic performed better in finding a feasible solution (i.e., categorized as either optimal or feasible in Table 5.3) within the time limit, while the exact method performed much better in finding an optimal solution.

The methods' ability to find a feasible solution within the time limit depends on the instance's sparsity and size. Of the 26 instances categorized as inconclusive by the exact method, 25 are sparse, and 22 are large (and sparse). Note that 23 of these 26 instances were solved by the heuristic. Further, among the 24 instances categorized as inconclusive by the heuristic, 20 are sparse and 23 are small. Taken as a whole, these results show that even though both methods were able to solve many of the sparse instances, some sparse instances were computationally challenging (i.e., produced inconclusive results); while large and sparse instances are challenging for the exact method (22 of its 26 inconclusive outcomes), small and sparse instances are challenging for the heuristic (19 of its 24 inconclusive outcomes).

Applying the heuristic presents a trade-off between the cost in the compactness objective and the advantage in computational time. To quantify the overall computational time advantage of the heuristic over the exact method for each instance, we compute the ratio of the exact method's time to the heuristic's time. Then, we calculate the geometric mean of the ratios across the instances. Further, we compute the *approximation gap* for each instance to measure the cost in the compactness objective. Let OPT and $HEUR$ be the compactness of the exact and heuristic solutions, respectively. The approximation gap, quantified by $(HEUR - OPT)/OPT$, measures the relative compromise of the heuristic's compactness objective compared to the optimal compactness. If the exact method terminated without an optimal solution, the best known lower bound is used in place of OPT, in which case this gap serves as an upper bound to the heuristic's approximation gap. We focus on the 205 (and 44 large) instances for which both methods found a feasible solution. On average, in these and in the large instances, the heuristic is 3.80 and 13.86 times faster than the exact method, respectively, while exhibiting an average approximation gap of 30.07% and 38.46%, respectively. Hence, the computational time advantage of the heuristic over the exact method is more evident in solving large instances than all instances, even though a relatively high approximation gap accompanies this advantage.

While the heuristic provides a computational advantage on average, the exact method outperforms it in small instances. The exact method is faster than the heuristic in 80 out of all 252 instances, all of which are small. Among the 66 large instances, the heuristic is faster in 64 instances. This observation aligns with the earlier finding that the heuristic finds more feasible solutions among the larger instances compared to the exact method. Hence, if computational time is prioritized over achieving optimality, we recommend using the exact method for small instances and the heuristic for large instances.

5.6.4 Results for HCGP with $Q \in \{3, 4\}$

In this section, we investigate the performance of the branch-and-cut method for HCGP with $Q \in \{3, 4\}$ applied to the 19 dense graphs; sparse graph instance results are not presented here since they yielded infeasibility certificates for $Q \in \{3, 4\}$. Similar to the settings used in the previous section, we solved for three different values of $K \in \{2, 3, 4\}$, and two balance settings, $\tau = \infty$ and $\tau = 0.1$. In total, we solved 228 instances, with each run of the method given a time limit of one hour.

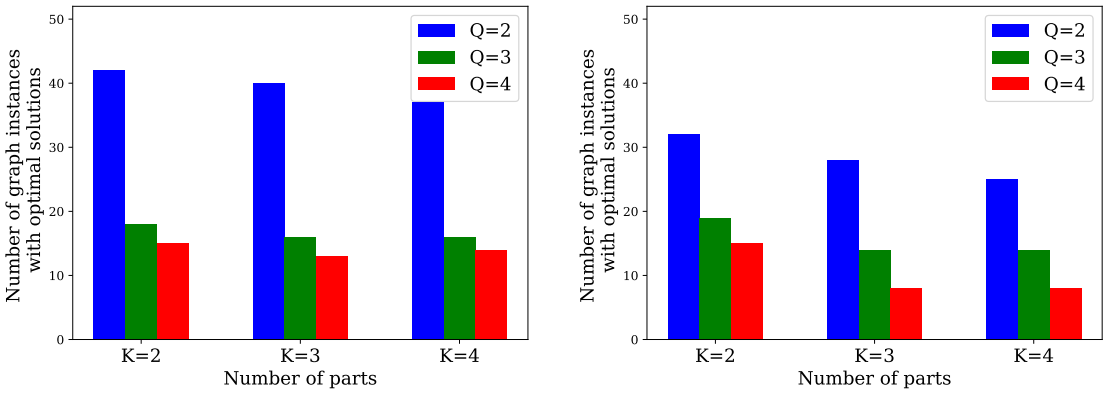
Table D.4 in Appendix D.3.2 presents the results from solving these 228 instances. The method terminated with an optimal (feasible) solution in 171 (192) instances. In 17 instances, the method produced an infeasibility certificate. In 19 instances, the method was inconclusive within the allotted time limit. Among the 17 infeasible instances, two (15) instances are for $Q = 3$ (4). None of these graph instances produced an infeasibility certificate for $Q = 2$. This trend demonstrates that the feasible region of HCGP becomes smaller when the required connectivity is increased from two through four.

5.6.5 Costs of Higher Connectivity

Ensuring highly connected partitions potentially incurs two costs: (i) a compromise in the optimal compactness objective due to the reduction in the solution space of feasible partitions, and (ii) an increase in computational time required to solve the associated integer program. In this analysis, we examine the optimal compactness value and the computational time for $Q \in \{2, 3, 4\}$ and contrast them with the baseline case of $Q = 1$. This comparison provides insight into the costs of ensuring each additional level of connectivity (beyond one).

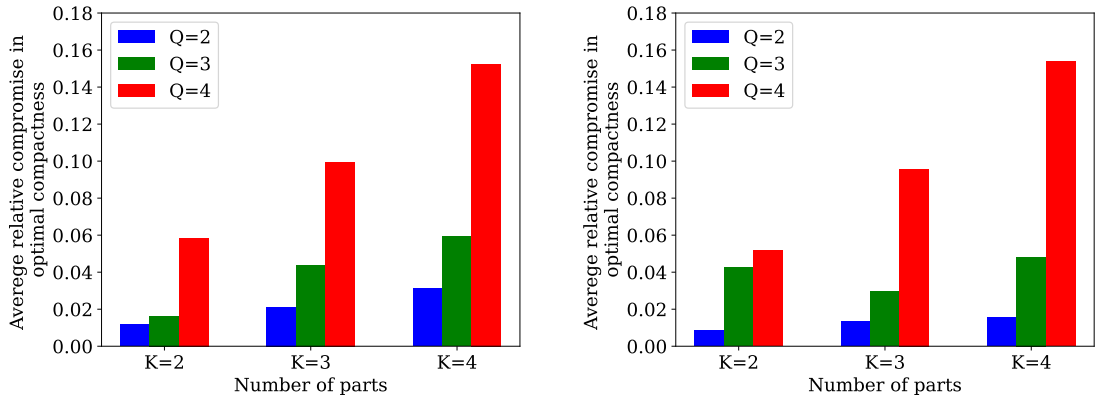
To quantify the baseline for these comparisons, Table D.5 in Appendix D.3.2 presents the results for $Q = 1$ across the 252 instances solved in Section 5.6.3 with a time limit of one hour. The branch-and-cut method terminated with an optimal (feasible) solution in 231 (236) instances. In 16 instances, the method was inconclusive; all of these instances are sparse graphs. None of these instances yielded an infeasibility certificate.

In our analysis, we compare the optimal compactness values and the computational times for each value of $Q \in \{2, 3, 4\}$ with the same for $Q = 1$. For a given $Q \in \{2, 3, 4\}$, let $\mathcal{I}_{1,Q}$ be the subset of instances for which the branch-and-cut method found an optimal solution for both $Q = 1$ and for the given Q. For each $Q \in \{2, 3, 4\}$, Figure 5.7 depicts the number of instances in $\mathcal{I}_{1,Q}$, categorized by K and τ. Furthermore, for each instance, let OPT_Q denote the optimal compactness for HCGP with $Q \geq 1$. For every $Q \in \{2, 3, 4\}$, we measure the *relative compromise* in optimal compactness from the baseline $Q = 1$ case as $(OPT_Q - OPT_1)/OPT_1$. Figure 5.8 depicts the average relative compromise in optimal compactness (among the instances in $\mathcal{I}_{1,Q}$ for $Q \in \{2, 3, 4\}$), categorized by K and τ. Additionally, to quantify the effect of Q on the computational time, we calculate the ratio between the computational times for each $Q \in \{2, 3, 4\}$ and for $Q = 1$. Figure 5.9 presents the geometric mean of the ratios (among the instances in $\mathcal{I}_{1,Q}$ for $Q \in \{2, 3, 4\}$), categorized by K and τ.



(a) $\tau = \infty$. (b) $\tau = 0.1$.

Figure 5.7: Number of instances in which the branch-and-cut method found optimal solutions for $Q = 1$ and subsequently found optimal solutions for each $Q \in \{2, 3, 4\}$; categorized for each $K \in \{2, 3, 4\}$ and balance $\tau \in \{0.1, \infty\}$.



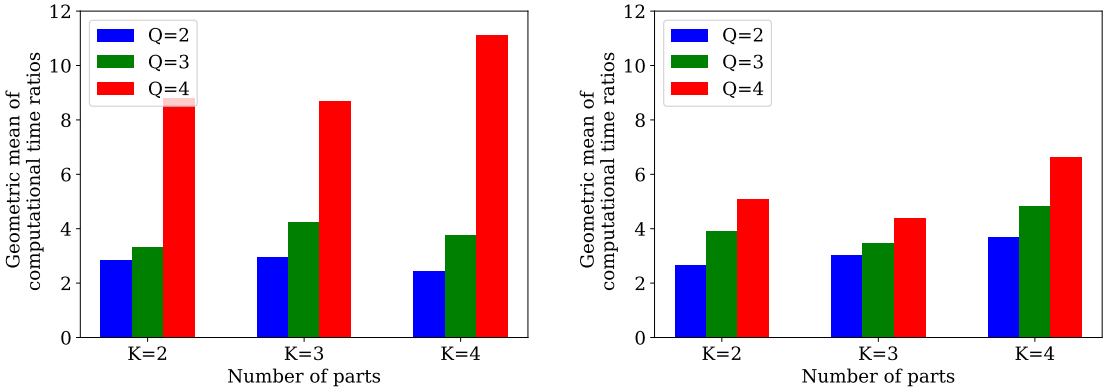
(a) $\tau = \infty$. (b) $\tau = 0.1$.

Figure 5.8: Average relative compromise in optimal compactness for $Q \in \{2, 3, 4\}$ compared to the baseline of $Q = 1$; categorized for each $K \in \{2, 3, 4\}$ and balance $\tau \in \{0.1, \infty\}$.

These empirical results suggest that increasing the required connectivity level Q leads to more infeasible instances, while also eroding the optimal compactness objective and increasing computational time among the feasible instances. For example, for $Q = 4$, $K = 4$ and $\tau = \infty$, on average among the 14 instances in $\mathcal{I}_{1,4}$, an optimal 4-proper partition exhibits a relative compromise in compactness objective of 15.24%, and requires approximately 11.2 times the computational effort compared to finding an optimal 1-proper partition.

5.6.6 Visual Examples

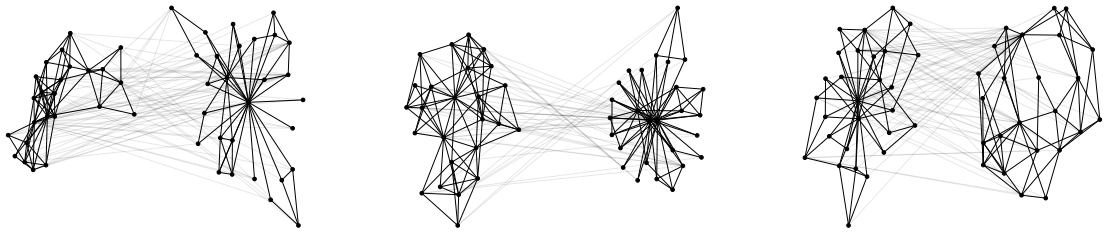
Finally, we present sample visualizations of optimal solutions from two graph instances. Figure 5.10 illustrates optimally compact and balanced Q-proper 2-partitions of a social network comprising 52 vertices for $Q \in [3]$. For $Q = 1$, notice the presence of a cut vertex in one of the parts. In the context of community detection in social networks, a partition with higher connectivity represents greater cohesion and robustness with multiple



(a) $\tau = \infty$. (b) $\tau = 0.1$.

Figure 5.9: Geometric mean of computational time ratios for $Q \in \{2, 3, 4\}$ compared to the baseline of $Q = 1$; categorized for each $K \in \{2, 3, 4\}$ and balance $\tau \in \{0.1, \infty\}$.

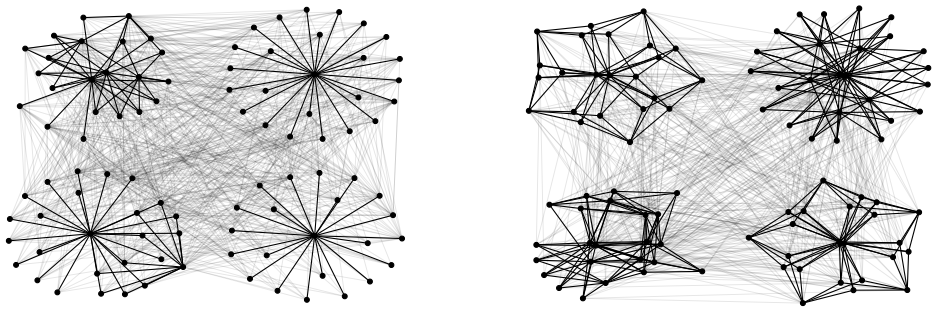
connections across the vertices within each part, as observed in the $Q = 3$ case.



(a) $Q = 1$. (b) $Q = 2$. (c) $Q = 3$.

Figure 5.10: Optimally compact and balanced (with $\tau = 0.1$) 2-partitions of a small social network with 52 vertices and 201 edges into Q-connected subgraphs for $Q \in [3]$; each part has a Kamada Kawai graph embedding.

Next, Figure 5.11 illustrates optimally compact and balanced Q-proper 4-partitions of a Mycielskian graph with 95 vertices for $Q \in \{1, 4\}$. This graph has the special property that it is triangle-free and has a chromatic number of seven. A graph with a high chromatic number has a complex structure where the vertices are interconnected so that they cannot be colored with a small number of colors without violating the rule of adjacent vertices having different colors. Observe that the 1-proper partition includes two parts with star-like structures, wherein removing the central vertex from the part would disconnect that part into singleton components. Hence, a vertex disruption to the central vertex would maximally degrade that part. In contrast, the 4-proper partition exhibits a higher level of connectivity across its parts, thereby being more resilient to vertex failures.



(a) $Q = 1$.

(b) $Q = 4$.

Figure 5.11: Optimally compact and balanced (with $\tau = 0.1$) Q-proper 4-partitions of a Mycielskian graph with 95 vertices and 755 edges for $Q \in \{1, 4\}$; each part has a Kamada Kawai graph embedding.

5.7 Conclusions

In summary, this study combines two fundamental graph concepts of graph partitioning and vertex connectivity. This paper presents an IP-based formulation for HCGP and solution methods to solve it. Through computational analysis, we evaluate the performance of these methods across a diverse set of instances. The results showcase the effectiveness of the proposed methods in generating optimal or feasible solutions for HCGP. The analysis presented in this work serves as a starting point for studying the interconnection between graph partitioning and vertex connectivity.

This work lays the foundation for several potential directions for future research. First, investigating alternative formulations for the Q-connectivity constraints, such as using multi-commodity flows, could provide alternative ways to model and solve HCGP. Second, investigating new techniques to improve the efficiency of the solution methods could enable practitioners to solve large instances of HCGP. For example, variable fixing techniques such as in Validi et al., 2022 could improve the branch-and-cut method's efficiency. For the heuristic setting, a constructive procedure to generate a Q-connected partition for $Q \geq 3$ could be explored. Third, studying the implications of adopting HCGP for practical applications would be valuable, while incorporating application-specific objectives and constraints. For example, partitioning a power network typically incorporates constraints that regulate the amount of power transmitted within each part (Sun et al., 2016). Hence, tailoring HCGP for application-specific needs would contribute to a positive impact of HCGP in practice.

Chapter 6

Conclusions

This dissertation studies optimization methods in political districting with fairness considerations and graph partitioning with resilience requirements. The four parts of this dissertation provide mathematical models and computational tools that aid practical implementation.

For political districting, Chapter 2 extends existing compactness-focused optimization models to incorporate political fairness objectives. Mathematical formulations are provided to optimize efficiency gap, partisan asymmetry, and competitiveness, shedding light on the trade-offs between compactness and these political fairness objectives. While these models do not capture all practical redistricting requirements, they serve as a proof-of-concept for integrating fairness-based optimization models into political districting.

The Arizona case study in Chapter 3 focuses on adapting an optimization approach for practical redistricting. The computational challenge posed by large input sizes is addressed using a multilevel algorithm with local search. The multi-stage approach ensures that all six legal criteria in Arizona's constitution are captured to the extent possible, making it the first optimization tool that satisfies all legal criteria in a state. This case study advances the potential for using optimization tools for practical districting, particularly when a central decision maker (e.g., an independent redistricting commission) draws districts.

As an alternative to a central map-drawing process, Chapter 4 explores game-based map-drawing approaches that consider political parties as players who collectively create a district map. In the bisection and I-cut-you-freeze protocols, each player's turn is modeled as an optimization problem where their strategy is informed by optimal strategies for a simplistic (i.e., continuous non-geometric) setting. A computational case study in Iowa shows that the resulting maps are fairer than enacted congressional district maps.

Beyond political districting, Chapter 5 tackles the problem of incorporating resilience into graph partitioning and formalizes the Highly Connected Graph Partitioning (HCGP) problem. This work introduces an optimization model for HCGP and an efficient branch-and-cut method derived from contiguity-enforcement models used in political districting. For the special case of requiring 2-connectivity, a heuristic method is presented, deriving from the structure of 2-connected graphs. Computational results using instances from various domains (such as social, transportation, and power networks) provide insights into the performance of these methods in finding optimal and feasible solutions. These models and analyses advance our understanding of the intersection between graph partitioning and vertex connectivity.

References

[1] A. I. Abramowitz, B. Alexander, and M. Gunning. "Incumbency, Redistricting, and the Decline of Competition in U.S. House Elections". In: *The Journal of Politics* 68.1 (2006), pp. 75–88.

[2] W. P. Adams and H. D. Sherali. "Linearization Strategies for a Class of Zero-One Mixed Integer Programming Problems". In: *Operations Research* 38.2 (1990), pp. 217–226.

[3] N. Ahn and S. Park. "An optimization algorithm for the minimum k-connected m-dominating set problem in wireless sensor networks". In: *Wireless Networks* 21 (2015), pp. 783–792.

[4] R. Ainsworth, E. G. Munoz, and A. M. Gomez. "District competitiveness increases voter turnout: evidence from repeated redistricting in North Carolina". In: *University of Florida* (2022).

[5] AIRC. *Arizona Independent Redistricting Commission Hub*. [Accessed: 24-Dec-2022]. 2021. URL: https://redistricting-irc-az.hub.arcgis.com/pages/draft-maps#background.

[6] AIRC. *Communities of Interest Report: Examining Publicly-Submitted Data*. [Accessed: 11-May-2022]. 2021. URL: https://storymaps.arcgis.com/stories/962c25f0866e49c9bb8751831678524b.

[7] T. A. Aleinikoff and S. Issacharoff. "Race and redistricting: Drawing constitutional lines after Shaw v. Reno". In: *Mich. L. Rev.* 92 (1993), p. 588.

[8] Arizona. *Arizona State Constitution, Article 4, Part 2, Section 1*. [Accessed: 1-Oct-2021]. 2021. URL: https://www.azleg.gov/const/4/1.p2.htm.

[9] Arizona Daily Star. *Plan for Arizona's new congressional districts: 4 Republican, 3 Democratic, 2 "competitive"*. [Accessed: 21-Nov-2022]. 2021. URL: https://tucson.com/news/local/plan-for-arizonas-new-congressional-districts-4-republican-3-democratic-2-competitive/article_3187991a-62b1-11ec-82dc-6ffea552ff06.html.

[10] M. Avci and S. Topaloglu. "An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries". In: *Computers & Industrial Engineering* 83 (2015), pp. 15–29.

[11] F. Bacao, V. Lobo, and M. Painho. "Applying genetic algorithms to zone design". In: *Soft Computing* 9.5 (2005), pp. 341–348.

[12] Baker v. Carr. *Baker v. Carr*. 369 U.S. 186, U.S. Supreme Court, 1962.

[13] Ballotpedia. *Majority-minority districts*. [Accessed: 1-Nov-2021]. 2021. URL: https://ballotpedia.org/Electoral_system#Congressional_elections.

[14] Ballotpedia. *Missouri Senate Joint Resolution No. 38*. [Accessed: 28-Jan-2021]. 2020. URL: https://legiscan.com/MO/text/SJR38/2020.

[15] Ballotpedia. *Redistricting*. [Accessed: 1-Nov-2021]. 2021. URL: https://ballotpedia.org/Redistricting.

[16] Ballotpedia. *State-by-state redistricting procedures*. [Accessed: 1-Nov-2021]. 2020. URL: https://ballotpedia.org/State-by-state_redistricting_procedures.

[17] A. Becker et al. "Computational redistricting and the voting rights act". In: *Election Law Journal: Rules, Politics, and Policy* 20.4 (2021), pp. 407–441.

[18] P. Bedi and C. Sharma. "Community detection in social networks". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6.3 (2016), pp. 115–135.

[19] P. Belotti and S. Buchanan A.and Ezazipour. *Political districting to optimize the Polsby-Popper compactness score*. [Accessed: 31-Aug-2023]. 2023. URL: https://doc.arcgis.com/en/redistricting/review/viewing-compactness-tests.htm.

[20] G. Benadè, A. D. Procaccia, and J. Tucker-Foltz. "You can have your cake and redistrict it too". In: *ACM Transactions on Economics and Computation* (2021).

[21] M. Bernstein and M. Duchin. "A formula goes to court: Partisan gerrymandering and the efficiency gap". In: *Notices of the AMS* 64.9 (2017), pp. 1020–1024.

[22] C. E. Bichot and P. Siarry, eds. *Graph Partitioning*. John Wiley & Sons, Inc., 2013.

[23] J. Birge. "Redistricting to maximize the preservation of political boundaries". In: *Social Science Research* 12.3 (1983).

[24] I. Bliznets and N. Karpov. "Parameterized algorithms for partitioning graphs into highly connected clusters". In: *arXiv preprint arXiv:1706.09487* (2017). [Accessed: 12-June-2023].

[25] V. Borozan et al. "Partitioning a graph into highly connected subgraphs". In: *Journal of Graph Theory* 82.3 (2016), pp. 322–333.

[26] B. Bozkaya, E. Erkut, and G. Laporte. "A tabu search heuristic and adaptive memory procedure for political districting". In: *European Journal of Operational Research* 144.1 (2003), pp. 12–26.

[27] M. H. Browdy. "Simulated annealing: An improved computer model for political redistricting". In: *Yale Law & Policy Review* 8.1 (1990), pp. 163–179.

[28] A. Buchanan et al. "An integer programming approach for fault-tolerant connected dominating sets". In: *INFORMS Journal on Computing* 27.1 (2015), pp. 178–188.

[29] A. Buluç et al. "Recent advances in graph partitioning". In: *Algorithm Engineering*. Springer, 2016, pp. 117–158.

[30] S. Cannon et al. "Voting rights, Markov chains, and optimization by short bursts". In: *Methodology and Computing in Applied Probability* 25.1 (2023), p. 36.

[31] C. P. Chambers, A. D. Miller, and J. Sobel. "Flaws in the efficiency gap". In: *Journal of Law & Politics* 33 (2017), p. 1.

[32] T. Chatterjee et al. "On theoretical and empirical algorithmic analysis of the efficiency gap measure in partisan gerrymandering". In: *Journal of Combinatorial Optimization* 40 (2020), pp. 512–546.

[33] J. Chen and J. Rodden. "Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures". In: *Quarterly Journal of Political Science* 8 (2013), pp. 239–269.

[34] G. Chiandussi et al. “Comparison of multi-objective optimization methodologies for engineering applications”. In: *Computers & Mathematics with Applications* 63.5 (2012), pp. 912–942.

[35] Jeanne N Clelland et al. *Compactness statistics for spanning tree recombination.* [Accessed: 23-July-2023]. 2021. URL: arXivpreprintarXiv:2305.17298.

[36] F. Dai and J. Wu. “On constructing k-connected k-dominating set in wireless ad hoc and sensor networks”. In: *Journal of Parallel and Distributed Computing* 66.7 (2006), pp. 947–958.

[37] S. J. D’Amico et al. “A simulated annealing approach to police district design”. In: *Computers & Operations Research* 29.6 (2002), pp. 667–684.

[38] T. A. Davis and Y. Hu. “The University of Florida Sparse Matrix Collection”. In: *ACM Transactions on Mathematical Software* 38.1 (2011). [Accessed: 15-March-2023], pp. 1–12. DOI: 10.1145/2049662.2049663.

[39] J. De Silva et al. “An analysis of a fair division protocol for drawing legislative districts”. In: *arXiv preprint arXiv:1811.05705* (2018).

[40] D. DeFord and M. Duchin. “Redistricting reform in Virginia: Districting criteria in context”. In: *Virginia Policy Review* 12.2 (2019), pp. 120–146.

[41] D. DeFord, M. Duchin, and J. Solomon. “A computational approach to measuring vote elasticity and competitiveness”. In: *Statistics and Public Policy* 7.1 (2020), pp. 69–86.

[42] D. DeFord, M. Duchin, and J. Solomon. “Recombination: A Family of Markov Chains for Redistricting”. In: *Issue 3.1, Winter 2021* (2021). DOI: 10.1162/99608f92.eb30390f.

[43] D. DeFord et al. “Implementing partisan symmetry: Problems and paradoxes”. In: *Political Analysis* 31.3 (2021), pp. 1–20.

[44] R. Diekmann et al. “Shape-optimized mesh partitioning and load balancing for parallel adaptive FEM”. In: *Parallel Computing* 26.12 (2000), pp. 1555–1581.

[45] O. Divounguy, J. Josko, and A. Schuster. *Competitive elections raise voter participation, uncontested elections hinder democracy.* [Accessed: 5-October-2023]. 2023. URL: https://www.illinoispolicy.org/reports/competitive-elections-raise-voter-participation-uncontested-elections-hinder-democracy/.

[46] K. W. Dobbs et al. “An Optimization Case Study in Analyzing Missouri Redistricting”. In: *INFORMS Journal on Applied Analytics* forthcoming (2023).

[47] A. Drexl and K. Haase. “Fast approximation methods for sales force deployment”. In: *Management Science* 45.10 (1999), pp. 1307–1323.

[48] M. Duchin. “Gerrymandering metrics: How to measure? What’s the baseline?” In: *arXiv preprint arXiv:1801.02064* (2018).

[49] M. Duchin and B. E. Tenner. “Discrete geometry for electoral geography”. In: (2018). [Accessed: 16-Sept-2023]. URL: https://arxiv.org/abs/1808.05860.

[50] J. Duda. *Republicans hold the edge as Arizona redistricting nears completion.* [Accessed: 21-Nov-2022]. 2021. URL: https://www.azmirror.com/2021/12/17/republicans-hold-the-edge-as-arizona-redistricting-nears-completion/.

[51] J. Edmonds. “Paths, trees, and flowers”. In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467.

[52] U. Elsner. *Graph Partitioning — A Survey*. Tech. rep. Technische Universitat Chemnitz, 1997.

[53] D. Eppstein. *Find maximum cardinality matching in general undirected graph*. [Accessed: 5-May-2020]. 2003. URL: https://www.ics.uci.edu/~eppstein/PADS/CardinalityMatching.py.

[54] D. Epstein and S. O'Halloran. "A social science approach to race, redistricting, and representation". In: *American Political Science Review* 93.1 (1999), pp. 187–191.

[55] A. H. Esfahanian. "Connectivity algorithms". In: *Topics in Structural Graph Theory* (2013), pp. 268–281.

[56] ESRI. *Efficiency gap (measure of partisan gias in redistricting) by state, 2016*. [Accessed: 27-Oct-2021]. 2021. URL: https://www.arcgis.com/home/item.html?id=563f61cd84a344deb707eed3e0b258bd.

[57] ESRI. *Viewing compactness tests*. [Accessed: 2-Sept-2023]. 2021. URL: https://doc.arcgis.com/en/redistricting/review/viewing-compactness-tests.htm.

[58] C. Fan et al. "A spatiotemporal compactness pattern analysis of congressional districts to assess partisan gerrymandering: a case study with California and North Carolina". In: *Annals of the Association of American Geographers* 105.4 (2015), pp. 736–753.

[59] M. Ferrara, C. Magnant, and P. Wenger. "Conditions for families of disjoint k-connected subgraphs in a graph". In: *Discrete Mathematics* 313.6 (2013), pp. 760–764.

[60] B. Fifield et al. "A new automated redistricting simulator using Markov chain Monte Carlo". In: *Working Paper, Princeton University, Princeton, NJ* (2015). [Accessed: 25-Dec-2020]. URL: http://imai.princeton.edu/research/files/redist.pdf.

[61] M. Fischetti et al. "Thinning out Steiner trees: a node-based model for uniform edge costs". In: *Mathematical Programming Computation* 9.2 (2017), pp. 203–229.

[62] J. Fravel et al. *Dual Bounds for Redistricting Problems with Non-Convex Objectives*. [Accessed: 31-Aug-2023]. 2023. URL: arXivpreprintarXiv:2305.17298.

[63] J. N. Friedman and R. T. Holden. "The rising incumbent reelection rate: What's gerrymandering got to do with it?" In: *The Journal of Politics* 71.2 (2009), pp. 593–611.

[64] H. N. Gabow and R. E. Tarjan. "Algorithms for two bottleneck optimization problems". In: *Journal of Algorithms* 9.3 (1988), pp. 411–417.

[65] M. R. Garey and D. S. Johnson. ""Strong" NP-completeness results: Motivation, examples, and implications". In: *Journal of the ACM (JACM)* 25.3 (1978), pp. 499–508.

[66] M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problems". In: *Theoretical Computer Science* 1.3 (1976), pp. 237–267.

[67] R. S. Garfinkel and G. L. Nemhauser. "Optimal political districting by implicit enumeration techniques". In: *Management Science* 16.8 (1970), B–495.

[68] S. Gentry et al. "Gerrymandering for justice: Redistricting U.S. liver allocation". In: *Interfaces* 45.5 (2015), pp. 462–480.

[69] J. A. George, B. W. Lamar, and C. A. Wallace. "Political district determination using large-scale network optimization". In: *Socio-Economic Planning Sciences* 31.1 (1997), pp. 11–28.

[70] Gerrymandering Project. *Redistricting Report Card*. [Accessed: 27-July-2023]. 2023. URL: https://gerrymander.princeton.edu/redistricting-report-card.

[71] M. Ghavidelsyooki et al. “Partitioning of transportation networks under disruption”. In: *International Journal of Modelling and Simulation* 37.3 (2017), pp. 131–139.

[72] Gill v. Whitford. *Gill v. Whitford*. Case 16-1161, U.S. Supreme Court, 2018.

[73] N. Goedert et al. *Black representation and district compactness in southern congressional districts*. [Accessed: 16-Aug-2023]. 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4449256.

[74] S. C. Gordon and G. Huber. “The effect of electoral competitiveness on incumbent behavior”. In: *Quarterly Journal of Political Science* 2.2 (2007), pp. 107–138.

[75] B. Grimstad and B. R. Knudsen. “Mathematical programming formulations for piecewise polynomial functions”. In: *Journal of Global Optimization* 77.3 (2020), pp. 455–486.

[76] B. Grofman. “Measures of bias and proportionality in seats-votes relationships”. In: *Political Methodology* (1983), pp. 295–327.

[77] B. Grofman and J. Cervas. “Recent Approaches to the Definition and Measurement of Compactness”. In: *SSRN* (2021). URL: https://ssrn.com/abstract=3919249.

[78] B. Grofman and J. R. Cervas. “The Terminology of Districting”. In: *Available at SSRN 3540444* (2020). URL: https://dx.doi.org/10.2139/ssrn.3540444.

[79] B. Grofman and G. King. “The future of partisan symmetry as a judicial test for partisan gerrymandering after LULAC v. Perry”. In: *Election Law Journal* 6.1 (2007), pp. 2–35.

[80] W. Gurnee and D. B. Shmoys. “Fairmandering: A column generation heuristic for fairness-optimized political districting”. In: *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*. SIAM. 2021, pp. 88–99.

[81] K. Haase and S. Müller. “Upper and lower bounds for the sales force deployment problem with explicit contiguity constraints”. In: *European Journal of Operational Research* 237.2 (2014), pp. 677–689.

[82] I. A. Hamad, P. A. Rikvold, and S. V. Poroseva. “Floridian high-voltage power-grid network partitioning and cluster optimization using simulated annealing”. In: *Physics Procedia* 15 (2011), pp. 2–6.

[83] E. Hartuv and R. Shamir. “A clustering algorithm based on graph connectivity”. In: *Information Processing Letters* 76.4-6 (2000), pp. 175–181.

[84] B. Hendrickson and T. G. Kolda. “Graph partitioning models for parallel computing”. In: *Parallel Computing* 26.12 (2000), pp. 1519–1534.

[85] B. Hendrickson and R. W. Leland. “A Multi-Level Algorithm For Partitioning Graphs.” In: *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing* 95.28 (1995).

[86] S. W. Hess et al. “Nonpartisan political redistricting by computer”. In: *Operations Research* 13.6 (1965), pp. 998–1006.

[87] S. Hirano and J. M. Snyder. “Primary elections and political accountability: What happens to incumbents in scandals?” In: *Quarterly Journal of Political Science* 7.4 (2012), pp. 447–456.

[88] ICSEG. *IEEE 300-Bus System*. https://icseg.iti.illinois.edu/ieee-300-bus-system/. 2013.

[89] S. Jackman. “Measuring electoral bias: Australia, 1949–93”. In: *British Journal of Political Science* 24.3 (1994), pp. 319–357.

[90] P. E. Jones. "The effect of political competition on democratic accountability". In: *Political Behavior* 35.3 (2013), pp. 481–515.

[91] D. Joshi, J. Soh, and A. Samal. "Redistricting using constrained polygonal clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 24.11 (2012), pp. 2065–2079.

[92] J. Kalcsics and R. Z. Ríos-Mercado. "Districting problems". In: *Location Science* (2019), pp. 705–743.

[93] G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 359–392.

[94] J. N. Katz, G. King, and E. Rosenblatt. "Theoretical foundations and empirical evaluations of partisan fairness in district-based democracies". In: *American Political Science Review* 114.1 (2020), pp. 164–178.

[95] A. R. Kaufman, G. King, and M. Komisarchik. "How to Measure Legislative District Compactness If You Only Know It When You See It". In: *American Journal of Political Science* 65.3 (2021), pp. 533–550.

[96] L. Kennedy, B. Corriher, and D. Root. *Redistricting and Representation*. [Accessed: 27-Oct-2021]. 2016. URL: https://www.americanprogress.org/issues/democracy/reports/2016/12/05/294272/redistricting-and-representation/.

[97] B. W. Kernighan and S. Lin. "An efficient heuristic procedure for partitioning graphs". In: *The Bell System Technical Journal* 49.2 (1970), pp. 291–307.

[98] D. M. King, S. H. Jacobson, and E. C. Sewell. "Efficient geo-graph contiguity and hole algorithms for geographic zoning and dynamic plane graph partitioning". In: *Mathematical Programming* 149.1-2 (2015), p. 425.

[99] D. M. King, S. H. Jacobson, and E. C. Sewell. "The geo-graph in practice: creating United States Congressional Districts from census blocks". In: *Computational Optimization and Applications* 69.1 (2018), pp. 25–49.

[100] D. M. King et al. "Geo-graphs: an efficient model for enforcing contiguity and hole constraints in planar graph partitioning". In: *Operations Research* 60.5 (2012), pp. 1213–1228.

[101] G. King et al. "Brief of Amici Curiae Professors Gary King, Bernard Grofman, Andrew Gelman, and Jonathan N. Katz, in Support of Neither Party". In: *U.S. Supreme Court in Jackson v. Perry* 05-204 (2006).

[102] T. Koch et al. "Progress in mathematical programming solvers from 2001 to 2020". In: *EURO Journal on Computational Optimization* 10 (2022), p. 100031.

[103] Z. Landau and F. E. Su. "Fair division and redistricting". In: *The Mathematics of Decisions, Elections, and Games. American Mathematical Society* 624 (2014), pp. 17–36.

[104] M. Lavrov. *Is there a polynomial time algorithm for min max maximal matching problem?* Mathematics Stack Exchange. [Accessed: 30-Nov-2019]. 2019. URL: https://math.stackexchange.com/q/3456687.

[105] J. Leskovec and A. Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. https://snap.stanford.edu/data/ego-Facebook.html. [Accessed: 3-November-2022]. 2014.

[106] J. Leskovec and J. Mcauley. "Learning to discover social circles in ego networks". In: *Advances in Neural Information Processing Systems* 25 (2012).

[107] H. A. Levin and S. A. Friedler. "Automated Congressional Redistricting". In: *Journal of Experimental Algorithmics* 24 (2019), pp. 1–24.

[108] Y. Li et al. "On the construction of k-connected m-dominating sets in wireless networks". In: *Journal of Combinatorial Optimization* 23 (2012), pp. 118–139.

[109] H. Liu et al. "Mathematical models of political districting for more representative governments". In: *Computers & Industrial Engineering* 140 (2020).

[110] Y. Y. Liu, W. K. T. Cho, and S. Wang. "PEAR: A massively parallel evolutionary computation approach for political redistricting optimization and analysis". In: *Swarm and Evolutionary Computation* 30 (2016), pp. 78–92.

[111] I. G. Ludden et al. "A bisection protocol for political redistricting". In: *INFORMS Journal on Optimization* 5.3 (2023), pp. 233–255.

[112] LULAC v. Perry. *LULAC v. Perry*. 548 U.S. 399, U.S. Supreme Court, 2006.

[113] D. B. Magleby and D. B. Mosesson. "A new approach for developing neutral redistricting plans". In: *Political Analysis* 26.2 (2018), pp. 147–167.

[114] T. E. Mann. "Redistricting Reform". In: *The National Voter* (2005). [Accessed: 25-Dec-2020]. URL: https://www.brookings.edu/articles/redistricting-reform/.

[115] N. McCarty, K. T. Poole, and H. Rosenthal. "Does gerrymandering cause polarization?" In: *American Journal of Political Science* 53.3 (2009), pp. 666–680.

[116] G. P. McCormick. "Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems". In: *Mathematical Programming* 10.1 (1976), pp. 147–175.

[117] M. P. McDonald. "Drawing the line on district competition". In: *PS: Political Science & Politics* 39.1 (2006), pp. 91–94.

[118] E. McGhee. "Partisan Gerrymandering and Political Science". In: *Annual Review of Political Science* 23 (2020), pp. 171–185.

[119] T. McGovern. *United States General Election Presidential Results by County from 2008 to 2016*. [Accessed: 25-Dec-2020]. 2017. URL: https://github.com/tonmcg/US_County_Level_Election_Results_08-16.

[120] N. Megiddo and K. J. Supowit. "On the complexity of some common geometric location problems". In: *SIAM Journal on Computing* 13.1 (1984), pp. 182–196.

[121] A. Mehrotra, E.L. Johnson, and G.L. Nemhauser. "An optimization based heuristic for political districting". In: *Management Science* 44.8 (1998), pp. 1100–1114.

[122] MGGG. *The geospatial toolkit for redistricting data.* [Accessed: 24-Oct-2021]. 2021. URL: https://github.com/mggg/maup.

[123] S. Micali and V. V. Vazirani. "An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs". In: *21st Annual Symposium on Foundations of Computer Science (SFCS 1980)*. IEEE. 1980, pp. 17–27.

[124] MIT Election Data and Science Lab. *County Presidential Election Returns 2000-2020*. [Accessed: 04-Feb-2022]. 2020. DOI: 10.7910/DVN/VOQCHQ. URL: https://doi.org/10.7910/DVN/VOQCHQ.

[125] N. Morgan and D. Howard. *Arizona redistricting and destiny: The 2022 results mostly match the 2021 expectations*. [Accessed: 24-Dec-2022]. 2022. URL: https://www.azmirror.com/2022/12/05/arizona-redistricting-and-destiny-the-2022-results-mostly-match-the-2021-expectations/.

[126] National Conference of State Legislatures. *Redistricting Criteria*. [Accessed: 25-Dec-2020]. 2020. URL: https://www.ncsl.org/research/redistricting/redistricting-criteria.aspx.

[127] R. G. Niemi et al. "Measuring compactness and the role of a compactness standard in a test for partisan and racial gerrymandering". In: *The Journal of Politics* 52.4 (1990), pp. 1155–1181.

[128] J. Oehrlein and J. H. Haunert. "A cutting-plane method for contiguity-constrained spatial aggregation". In: *Journal of Spatial Information Science* 15 (2017), pp. 89–120.

[129] Office of the Governor. *Executive Order 66: Relating to creating the people's maps commission*. The State of Wisconsin, 2019.

[130] B. Olson. *Redistricter*. [Accessed: 24-Dec-2022]. 2013. URL: https://code.google.com/archive/p/redistricter/.

[131] OpenPrecincts. *Arizona*. [Accessed: 08-Dec-2021]. 2020. URL: https://openprecincts.org/az.

[132] G. Orman. "Multiple communities of ego in social networks". In: *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering*. 2021, pp. 128–133.

[133] G. K. Orman and O. Karadeli. "Overlapping communities via k-connected ego centered groups". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. 2015, pp. 1598–1599.

[134] W. Pegden, A. D. Procaccia, and D. Yu. "A partisan districting protocol with provably nonpartisan outcomes". In: *arXiv preprint arXiv:1710.08781* (2017).

[135] D. A. Plane. "Redistricting reformulated: A maximum interaction/minimum separation objective". In: *Socio-Economic Planning Sciences* 16.6 (1982), pp. 241–244.

[136] D. D. Polsby and R. D. Popper. "The third criterion: Compactness as a procedural safeguard against partisan gerrymandering". In: *Yale Law and Policy Review* 9 (1991), p. 301.

[137] E. C. Reock. "A note: Measuring compactness as a requirement of legislative apportionment". In: *Midwest Journal of Political Science* 5.1 (1961), pp. 70–74.

[138] Reynolds v. Sims. *Reynolds v. Sims*. 377 U.S. 533, U.S. Supreme Court, 1964.

[139] F. Ricca and A. Scozzari. "Mathematical programming formulations for practical political districting". In: *Optimal Districting and Territory Design* 284 (2020), pp. 105–128.

[140] F. Ricca, A. Scozzari, and B. Simeone. "Political districting: From classical models to recent approaches". In: *Annals of Operations Research* 204.1 (2013), p. 271.

[141] F. Ricca, A. Scozzari, and B. Simeone. "Weighted Voronoi region algorithms for political districting". In: *Mathematical and Computer Modelling* 48.9-10 (2008), pp. 1468–1477.

[142] F. Ricca and B. Simeone. "Local search algorithms for political districting". In: *European Journal of Operational Research* 189.3 (2008), pp. 1409–1426.

[143] R. A. Rossi and N. K. Ahmed. "The Network Data Repository with Interactive Graph Analytics and Visualization". In: *AAAI*. [Accessed: 2-May-2023]. 2015. URL: https://networkrepository.com.

[144] L. Royden and M. Li. *Extreme maps*. [Accessed: 2-May-2018]. Brennan Center for Justice at New York University School of Law, 2017. URL: https://www.brennancenter.org/sites/default/files/publications/Extreme%20Maps%205.16_0.pdf.

[145] P. A. Rubin. *Optimizing part of the objective function II*. [Accessed: 27-Oct-2021]. 2015. URL: https://orinanobworld.blogspot.com/2015/08/optimizingpartoftheobjectivefunction-ii.html.

[146] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, and J. L. González-Velarde. "A bi-objective programming model for designing compact and balanced territories in commercial districting". In: *Transportation Research Part C: Emerging Technologies* 19.5 (2011), pp. 885–895.

[147] T. Shirabe. "Districting modeling with exact contiguity constraints". In: *Environment and Planning B: Planning and Design* 36.6 (2009), pp. 1053–1066.

[148] K. Silverberg. "Illegitimacy of the incumbent gerrymander". In: *Texas Law Review* 74 (1995), p. 913.

[149] H. D. Simon. "Partitioning of unstructured problems for parallel processing". In: *Computing Systems in Engineering* 2.2-3 (1991), pp. 135–148.

[150] B. Stabler, H. Bar-Gera, and E. Sall. *Transportation Networks for Research*. [Accessed: 5-November-2022]. 2020. URL: https://github.com/bstabler/TransportationNetworks.

[151] N. O. Stephanopoulos and E. M. McGhee. "Partisan gerrymandering and the efficiency gap". In: *The University of Chicago Law Review* 82.2 (2015), pp. 831–900.

[152] N. O. Stephanopoulos and E. M. McGhee. "The Measure of a Metric: The Debate over Quantifying Partisan Gerrymandering". In: *Stanford Law Review* 70.5 (2018).

[153] L. Sun et al. "Network partitioning strategy for parallel power system restoration". In: *IET Generation, Transmission & Distribution* 10.8 (2016), pp. 1883–1892.

[154] R. Swamy, D. M. King, and S. H. Jacobson. "Multiobjective Optimization for Politically Fair Districting: A Scalable Multilevel Approach". In: *Operations Research* 71.2 (2023), pp. 536–562.

[155] K. Tapp. "Measuring political gerrymandering". In: *The American Mathematical Monthly* 126.7 (2019), pp. 593–609.

[156] The Guardian. *U.S. 2012 election data*. [Accessed: 5-May-2018]. 2018. URL: https://www.theguardian.com/news/datablog/2012/nov/07/us-2012-election-county-results-download.

[157] J. D. Thoreson and J. M. Liittschwager. "Computers in behavioral science. Legislative districting by computer simulation". In: *Behavioral Science* 12.3 (1967), pp. 237–247.

[158] Thornburg v. Gingles. *Thornburg v. Gingles*. 478 U.S. 30, U.S. Supreme Court, 1986.

[159] A. Timofieiev, V. Snasel, and J. Dvorsky. "Social communities detection in Enron Corpus using h-Index". In: *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*. IEEE. 2008, pp. 507–512.

[160] U.S. Census Bureau. *Census Block Tallies by State or State Equivalent*. [Accessed: 27-Dec-2017]. 2010. URL: https://www.census.gov/data.html.

[161] U.S. Census Bureau. *Census Block Tallies by State or State Equivalent*. [Accessed: 3-May-2022]. 2020. URL: https://www.census.gov/data.html.

[162] H. Validi and A. Buchanan. "Political districting to minimize cut edges". In: *Mathematical Programming Computation* 14.4 (2022), pp. 623–672.

[163] H. Validi, A. Buchanan, and E. Lykhovyd. "Imposing contiguity constraints in political districting models". In: *Operations Research* 70.2 (2022), pp. 867–892.

[164] E. Veomett. "Efficiency gap, voter turnout, and the efficiency principle". In: *Election Law Journal: Rules, Politics, and Policy* 17.4 (2018), pp. 249–263.

[165] W. Vickrey. "On the prevention of gerrymandering". In: *Political Science Quarterly* 76.1 (1961), pp. 105–110.

[166] J. P. Vielma, S. Ahmed, and G. Nemhauser. "Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions". In: *Operations Research* 58.2 (2010), pp. 303–315.

[167] C. Walshaw. "A multilevel algorithm for force-directed graph drawing". In: *International Symposium on Graph Drawing*. Springer. 2000, pp. 171–182.

[168] C. Walshaw. "A multilevel approach to the travelling salesman problem". In: *Operations Research* 50.5 (2002), pp. 862–877.

[169] G. S. Warrington. "Quantifying gerrymandering using the vote distribution". In: *Election Law Journal* 17.1 (2018), pp. 39–57.

[170] Wesberry et al. v. Sanders, Governor of Georgia, et al. *Wesberry et al. v. Sanders, Governor of Georgia, et al.* Case 376 U.S. 1., No. 22., U.S. Supreme Court, 1964.

[171] H. Whitney. "Non-separable and planar graphs". In: *Proceedings of the National Academy of Sciences* 17.2 (1931), pp. 125–127.

[172] K. Wong. *2022 Elections: Did your Congressional district change this year? Here's the reason why.* [Accessed: 24-Dec-2022]. 2022. URL: https://www.fox10phoenix.com/news/arizona-congressional-districts-changed-heres-the-reason-why.

[173] H. P. Young. "Measuring the compactness of legislative districts". In: *Legislative Studies Quarterly* 13.1 (1988), pp. 105–115.

Appendix A

Appendix for Chapter 2

A.1 Debate on Efficiency Gap

Efficiency Gap (EG) has emerged as a prominent measure of partisan fairness due to its elegant definition and simplicity in computation, and has even been used by the Supreme Court in Wisconsin's partisan gerrymandering case (Gill v. Whitford, 2018). Recently, the state of Missouri passed a resolution that requires that the district plan drawn for the 2021 districting cycle must have an EG less than 15% (Ballotpedia, 2020a). In their seminal work introducing the efficiency gap, (Stephanopoulos and McGhee, 2015) propose an 8% threshold for EG beyond which a district plan can be considered "unfair" on partisan grounds.

The spotlight on EG has led to debate on its legitimacy as a measure of partisan fairness. Bernstein and Duchin, 2017, Chambers et al., 2017 and Veomett, 2018 provide valid criticisms on the theoretical efficacy of EG in assessing gerrymandering. Four of the key concerns are discussed here.

1. The computation of EG is deterministic, but it requires the use of uncertain and volatile election results. This factor is particularly critical in the presence of competitive districts, where a small change in voting behavior could "flip" the party holding the majority in those districts, thereby altering the value of EG drastically. To improve the robustness of computing EG, Stephanopoulos and McGhee, 2015 suggest using a sensitivity analysis in a second phase of the computation.

2. A plan with a small value of EG (i.e., considered "fair" according to EG) could inadvertently favor districts that are highly non-competitive, or are highly "packed". Such districts may be considered undesirable by a districting process that seeks to promote competition among the districts.

3. When districts have unequal voter turnouts, Veomett, 2018 shows that even a district plan with EG $= 0$ could favor one party disproportionately more than the other. This is possible in a scenario where the turnout in a few districts is disproportionately larger than the turnouts in other districts, and so more votes are wasted in the districts with high turnout. This discrepancy benefits the party that wins in the districts with lesser turnout. To illustrate this effect, Veomett, 2018 constructs a district plan with EG $= 0$, where a party with a 50% vote-share in the state disproportionately wins 60% of the districts. In this plan, the turnout in each district they lose is 1.5 times the turnout in each district they win.

4. The mathematical structure of EG exhibits certain undesirable properties when assessing district plans in states with highly polarized vote-shares. In particular, Bernstein and Duchin, 2017 derive that in a

state where one party has an overall vote-share greater than 80%, every district plan would have an EG value greater than 0.08, which is a threshold for EG suggested by Stephanopoulos and McGhee, 2015 beyond which a district plan may be considered to be gerrymandered.

In practice, none of the 50 states have a lean greater than 80% based on the 2016 presidential election. Here, lean refers to the overall vote-share of the majority party in the state. Even for the states with the highest leans, it is possible to achieve EG values less than 0.08. Among the districts with six or more districts, California, Kentucky and Massachusetts have the highest leans of 66.1%, 65.7% and 64.7%, respectively, while the EG values of their congressional district plans (as of 2016) are 0.004, 0.035 and 0.057, respectively (ESRI, 2021a). Further, EG correctly identifies partisan gerrymandering in practical instances. Michigan, North Carolina and Pennsylvania have the highest levels of partisan bias when measured using multiple quantitative measures (Royden and Li, 2017). Their EG values are 0.132, 0.203 and 0.160, respectively, even though their vote-shares are all under 52% (ESRI, 2021a). Hence, despite the theoretical limitations of EG, its performance as a measure of partisan gerrymandering is reasonably good in practice.

In their conclusion, Bernstein and Duchin, 2017 note that despite the flaws they raise, they "assess that EG can still be a useful component of a courtroom analysis". In their rebuttal, Stephanopoulos and McGhee, 2018 argue that despite these flaws, EG satisfies certain fundamental principles of a partisan fairness metric (such as *efficiency*, *distinctness*, and *breadth of scope*), which justifies its legitimacy as a measure of partisan fairness. See Stephanopoulos and McGhee, 2018 for more details about these principles.

A.2 Debate on Partisan Asymmetry

The symmetry standard as a measure of partisan fairness was put to test in LULAC v. Perry, 2006, a key U.S. Supreme Court gerrymandering case involving the 2003 Texas' legislative district plan. An amicus brief filed by King et al., 2006 proposed that the gerrymandering test be based on partisan asymmetry since "a consensus exists about using the symmetry standard" to evaluate partisan bias. The proposal was discussed, and even though three of the opinions positively evaluated the symmetry standard, the majority opinion had reservations with adopting the test (Grofman and King, 2007). Justice Kennedy expressed concerns with measuring symmetry through the *hypothetical* construction of vote-shares, noting that the court is "wary of adopting a constitutional standard that invalidates a plan based on unfair results that would occur in a hypothetical state of affairs" (LULAC v. Perry, 2006). Justice Stevens (joined by Justice Breyer) endorsed the view that "the proponents of the symmetry standard have provided a helpful (though certainly not talismanic) tool" to measure partisan fairness, though Justice Kennedy (joined by Justices Souter and Ginsburg) noted that "asymmetry alone is not a reliable measure of unconstitutional partisanship" (LULAC v. Perry, 2006). This view motivates the inclusion of partisan asymmetry as one objective in the multi-objective modeling framework for political districting presented in this chapter.

A plan that emphasizes symmetry may not emphasize proportionality. Even though proportionality is not a constitutional requirement, some authors consider this lack of synergy between symmetry and proportionality to be a flaw of the symmetry standard itself. For example, a recent technical report by DeFord et al., 2021b finds that when randomly simulating a large number of district plans in Utah, Texas and North Carolina (with Republican vote-shares of 72%, 58% and 53%, respectively), the district plans with small partisan asymmetry values lean disproportionally in favor of Republicans in most cases. Hence, these plans may be viewed as gerrymandered from the proportionality standpoint. In their conclusion, DeFord

et al., 2021b note that none of their findings "gives a theoretical reason for rejecting partisan symmetry as a definition of fairness" since a proponent of symmetry could argue that the symmetry standard reinforces "the legitimacy of district-based democracy by reassuring the voting public that the tables can yet turn in the future." In the case study on Wisconsin (with a Republican vote-share of 48.4%) presented in Section 2.5, it is found that the majority of the plans align with proportionality, suggesting that there are cases where symmetry does synergize with proportionality.

A.3 Proof for Theorem 1

Theorem 1. *For a general graph $G = (V, E)$, unit populations $\{p_i, p_i^A, p_i^B\}_{i \in V}$ and number of districts $K \geq 2$, the Aggregate Competitive Districting Problem (ACDP) with or without the population balance constraints is NP-complete if $K = 2$ and is strongly NP-complete if $K \geq 3$.*

Proof. The decision version of ACDP is defined as follows. The inputs are a graph $G = (V, E)$, unit populations $\{p_i, p_i^A, p_i^B\}_{i \in V}$, a balance threshold $\tau \geq 0$, number of districts $K \geq 2$, and a target $W \geq 0$. Then, the decision version of ACDP without population balance constraints asks if there exists a feasible district plan $z : V \to [K]$ such that

$$\max_{k \in [K]} |P_k^A(z) - P_k^B(z)| \leq W, \tag{A.1}$$

where for each district $k \in [K]$ and party $l \in \{A, B\}$, $P_k^l(z) = \sum_{i \in V : z(i) = k} p_i^l$. Here, a feasible district plan is one that satisfies the MIP constraints (2.3)-(2.9). For ACDP with population balance constraints, a feasible district plan also satisfies constraints (2.2) for a given $\tau \geq 0$. Given a district plan, since feasibility for ACDP with or without population balance constraints can be verified in polynomial time using the constraints (2.2)-(2.9) and (A.1), it is clear that ACDP with or without population balance constraints is in NP. It is now shown that ACDP is NP-complete for $K = 2$ districts, and is strongly NP-complete for $K \geq 3$ districts. A polynomial-time reduction is shown from the NP-complete 2-*partitioning problem* for $K = 2$ and from the strongly NP-complete 3-*partitioning problem* for $K \geq 3$ (Garey and Johnson, 1978). The two reductions are first shown for ACDP without the population balance constraints. Then, it is shown how a τ value can be chosen so that the reductions hold when the population balance constraints are added.

For $K = 2$ without population balance:

The 2−partitioning problem (or 2-PART) is defined as follows. For a given set of integers $V' = \{a_1, a_2 \ldots, a_n\}$ and a target $W' \geq 0$, 2-PART divides V' into two partitions $\{S_1, S_2\}$ such that $\max_{k=1,2} |\sum_{i \in S(k)} a_i| \leq W'$. For a given instance of 2-PART (V', W'), an instance of ACDP can be constructed such that solving the ACDP is equivalent to solving 2-PART. The set of units V is chosen to be the indices of the set of integers $[n]$, the number of districts K is equal to 2, and the adjacency graph G is constructed to be a complete graph. For every unit $i \in V$, the unit populations (p_i, p_i^A, p_i^B) are chosen to be $(a_i, a_i, 0)$. The target W is chosen to be W'. To show that the two problems are equivalent, it shown that the constructed instance of ACDP is a "yes" instance if and only if the given 2-PART instance is a "yes" instance.

First, let the instance of 2-PART be a "yes" instance, i.e., there exists a partition $\{S_1, S_2\}$ of V' such that $\max_{k=1,2} |\sum_{i \in S_k} a_i| \leq W'$. Consider a district plan z constructed by assigning units in V to the two districts according to the corresponding assignment of integers in the two sets S_1 and S_2. Then, for every

district $k = 1, 2$,

$$|P_k^A(z) - P_k^B(z)| = |\sum_{i \in V: z(i)=k} (p_i^A - p_i^B)| = |\sum_{i \in V'_k} (a_i - 0)| \leq W'.$$

Since $W = W'$, the district plan z satisfies constraint (A.1). Note that since G is a complete graph, z satisfies the contiguity constraints. Hence, the constructed instance to ACDP is a "yes" instance.

Second, let the constructed instance of ACDP be a "yes" instance, i.e., there exists a district plan z that satisfies constraints (2.3)-(2.9) and (A.1). Consider a partition of V' into 2 partitions $\{S_1, S_2\}$ constructed according to the assignment of the units in $V = V'$ to the $K = 2$ districts. For every $k = 1, 2$, as determined by z, $|\sum_{i \in S(k)} a_i| = |\sum_{i \in V: z(i)=k} (p_i^A - p_i^B)| \leq W$. Since $W = W'$, the partition $\{S_1, S_2\}$ satisfies $\max_{k=1,2} |\sum_{i \in S(k)} a_i| \leq W'$. Hence, the instance of 2-PART is a "yes" instance.

For $K \geq 3$ without population balance:

The 3−partitioning problem (or 3-PART) is defined as follows. For a given set of integers $V' = \{a_1, a_2 \ldots, a_n\}$ and a target $W' \geq 0$, 3-PART divides V' into three partitions $\{S_1, S_2, S_3\}$ such that $\max_{k=1,2,3} |\sum_{i \in S(k)} a_i| \leq W'$. For a given instance of 3-PART (V', W'), an instance of ACDP can be constructed such that solving the ACDP is equivalent to solving 3-PART. The number of districts is chosen to be $K \geq 3$. The set of units V is chosen to be the indices of the set of integers in V' padded with $K - 3$ additional units, i.e., $V = [n] \cup \{n + 1, n + 2, \ldots, n + K - 3\}$. The adjacency graph G is constructed to be a complete graph. For every unit $i \in V$, the unit populations (p_i, p_i^A, p_i^B) are chosen to be $(a_i, a_i, 0)$ if $i \in [n]$, or $(W', W', 0)$ if $n + 1 \leq i \leq n + K - 3$. The target W is chosen to be W'. To show that the two problems are equivalent, it is to be shown that an instance of ACDP is a "yes" instance if and only if the constructed 3-PART instance is a "yes" instance.

First, let the instance of 3-PART be a "yes" instance, i.e., there exists a partition $\{S_1, S_2, S_3\}$ of V' such that $\max_{k=1,2,3} |\sum_{i \in S(k)} a_i| \leq W'$. Consider a district plan z constructed by assigning units in $[n]$ to districts 1, 2 and 3 according to the corresponding assignment of integers in the three sets S_1, S_2, S_3. Each unit in $\{n + 1, n + 2, \ldots, n + K - 3\}$ is assigned to occupy a single district among the remaining districts in $[K] \backslash [3]$. Then, for every district $k = 1, 2, 3$,

$$|P_k^A(z) - P_k^B(z)| = |\sum_{i \in V: z(i)=k} (p_i^A - p_i^B)| = |\sum_{i \in V'_k} (a_i - 0)| \leq W'.$$

For every district $k \in K \backslash [3]$,

$$|P_k^A(z) - P_k^B(z)| = |\sum_{i \in V: z(i)=k} (p_i^A - p_i^B)| = |(W' - 0)| = W'.$$

Since $W = W'$, the district plan z satisfies constraint (A.1). Note that since G is a complete graph, z satisfies the contiguity constraints. Hence, the constructed instance to ACDP is a "yes" instance.

Second, let the instance of ACDP be a "yes" instance, i.e., there exists a district plan z that satisfies constraints (2.3)-(2.9) and (A.1). Note that each of the $K - 3$ padded units will have to be assigned to its own district in order to satisfy constraint (A.1). These units will occupy $K - 3$ of the K districts. A "yes" to 3-PART can be constructed using the units in $[n]$ to the remaining three districts; suppose, without loss of generality, that the three remaining districts correspond to districts $k = 1, 2, 3$. Consider a partition of V' into 3 partitions, $\{S_1, S_2, S_3\}$, constructed according to the assignment of the units in $[n]$ to the three districts. For every $k = 1, 2, 3$, $|\sum_{i \in S(k)} a_i| = |\sum_{i \in V: z(i)=k} (p_i^A - p_i^B)| \leq W$. Since $W = W'$, the partition $\{S_1, S_2, S_3\}$ satisfies $\max_{k=1,2,3} |\sum_{i \in S(k)} a_i| \leq W'$. Hence, the instance of 3-PART is a "yes" instance.

Inclusion of population balance constraints:

The population balance constraints (2.2) require that the population in every district is in the range $[\overline{P}(1-\tau), \overline{P}(1+\tau)]$, where $\overline{P} := (\sum_{i \in V} p_i)/K$. When these constraints are included in ACDP, it is needed to show that an instance of ACDP can be constructed for some $\tau \geq 0$ such that the two reductions in this proof hold true. Let $\tau = K - 1$. Then, the range of allowable district populations is $[\overline{P}(2-K), \overline{P}K]$ which is equivalent to $[0, \sum_{i \in V} p_i]$ for $K \geq 2$ since unit populations are non-negative. Then, every district plan will satisfy the population balance constraints, and the two reductions in this proof will continue to hold true. □

A.4 Proof for Proposition 1

Proposition 1. *Symmetric-DP can be formulated as a Mixed Integer Linear Program (MILP).*

Proof. To construct an MILP formulation for Symmetric-DP, each of the three constraints (2.23) - (2.25) are linearized one at a time in the following three parts.

(a) First, constraints (2.23) are linearized for given values of $\{x_{ij}\}_{i,j \in V}$, the decision variables that assign units to district centers. For each unit $i \in V$, the following set of constraints define ω_i to be the vote-share for party A in a district with center i; 0 if i is not a district center. Constraints (2.23) are given by,

$$\omega_i = \begin{cases} \dfrac{\sum_{j \in V} p_j^A x_{ij}}{\sum_{j \in V} (p_j^A + p_j^B) x_{ij}}, & \text{if } x_{ii} = 1 \\ 0, & \text{otherwise} \end{cases} \qquad \forall\ i \in V,$$

The following additional variables are defined.

$$\beta_{ij} = \begin{cases} \omega_i, & \text{if } i \in V \text{ is a district center, and } j \in V \text{ is assigned to that district.} \\ 0, & \text{otherwise .} \end{cases}$$

The first claim is that the following set of constraints linearize constraints (2.23).

$$\omega_i \leq x_{ii} \quad \forall\ i \in V, \tag{A.2}$$

$$\beta_{ij} \leq \omega_i \quad \forall\ i, j \in V, \tag{A.3}$$

$$\omega_i - 1 + x_{ij} \leq \beta_{ij} \quad \forall\ i, j \in V, \tag{A.4}$$

$$\beta_{ij} \leq x_{ij} \quad \forall\ i, j \in V, \tag{A.5}$$

$$\sum_{j \in V} (p_j^A + p_j^B) \beta_{ij} = \sum_{j \in V} p_j^A x_{ij} \quad \forall\ i \in V, \tag{A.6}$$

$$\beta_{ij}, \omega_i \geq 0 \quad \forall\ i, j \in V. \tag{A.7}$$

For every $i \in V$, constraint (A.2) ensures that if i is not a district center, then $\omega_i = 0$. For every $i, j \in V$, constraints (A.3)- (A.5) together establish the relationship $\beta_{ij} = \omega_i \cdot x_{ij}$. When $x_{ij} = 0$, constraint (A.5) ensures that $\beta_{ij} = 0$, and when $x_{ij} = 1$, constraints (A.3) and (A.4) ensure that $\beta_{ij} = \omega_i$. Given that $\beta_{ij} = \omega_i \cdot x_{ij}$, it can be seen that constraint (A.6) is equivalent to the non-linear constraint (2.23). Constraints (A.7) establish the non-negativity of the variables.

(b) Second, constraints (2.24) are linearized. Here, the values for $\{\omega_i\}_{i\in V}$ defined by constraints (A.2)-(A.7) are used to define α_k, the $k-$th largest district vote-share for $k \in [K]$ among the set of all district vote-shares. Constraints (2.24) are given by,

$$\alpha_k = k \max_{i\in V}\{\omega_i\} \quad \forall\ k \in [K],$$

where the k max function returns the $k-$th largest value of a given set. To linearize this constraint, for each $i \in V$ and $k \in [K]$, additional variables $\delta_{i,k}$ and $\gamma_{i,k}$ are now defined. Some additional notation is first introduced for each $k \in [K]$. Let $\omega_{(k)}$ denote the value of the $k-$th largest element in the set $\{\omega_i\}_{i\in V}$. For each $k \in [K]$, let $S^+(k), S(k), S(k)^- \subseteq V$ be subsets of units that partition V which are defined as $S^+(k) \coloneqq \{i \in V : \omega_i > \omega_{(k)}\}$, $S(k) \coloneqq \{i \in V : \omega_i = \omega_{(k)}\}$, and $S(k)^- \coloneqq \{i \in V : \omega_i < \omega_{(k)}\}$. It can be observed that $|S^+(k)| < k$ and $|S^-(k)| < |V| - k + 1$ since $|S(k)| \geq 1$ by definition. Among the units in $S(k) \cup S^+(k)$, choose an arbitrary subset of units $S'(k) \subseteq S(k) \cup S^+(k)$ such that $|S'(k)| = k$. Such an $S'(k)$ is possible since $|S(k) \cup S^+(k)| = |V| - |S^-(k)| > |V| - (|V| - k + 1) = k - 1$, and hence $|S(k) \cup S^+(k)| \geq k$ since the size of the set is an integer. Further, among the units in $S(k) \cup S^-(k)$, choose an arbitrary subset of units $S''(k) \subseteq S(k) \cup S^-(k)$ such that $|S''(k)| = |V| - k + 1$. Such an $S''(k)$ is possible since $|S(k) \cup S^-(k)| = |V| - |S^+(k)| > |V| - k$, and hence $|S(k) \cup S^-(k)| \geq |V| - k + 1$ since the size of the set is an integer. Based on the constructed sets $S'(k)$ and $S''(k)$, the $\delta_{i,k}$ and $\gamma_{i,k}$ values are defined as follows.

$$\delta_{i,k} = \begin{cases} 1, & \text{if } i \in S'(k), \\ 0, & \text{otherwise.} \end{cases} \tag{A.8}$$

$$\gamma_{i,k} = \begin{cases} 1, & \text{if } i \in S''(k), \\ 0, & \text{otherwise.} \end{cases} \tag{A.9}$$

For each $k \in [K]$, ensuring that α_k is the $k-$th largest element in the set $\{\omega_i\}_{i\in V}$ is equivalent to ensuring that α_k is less than or equal to at least k elements in the set, and that α_k is greater than or equal to at least $|V| - k + 1$ elements in the set. The second claim in this proof is that the following set of constraints linearize constraints (2.24).

$$\alpha_k \leq \omega_i + 1 - \delta_{i,k} \quad \forall\ i \in V,\ k \in [K] \tag{A.10}$$

$$\sum_{i\in V} \delta_{i,k} = k \quad \forall\ k \in [K] \tag{A.11}$$

$$\omega_i - 1 + \gamma_{i,k} \leq \alpha_k \quad \forall\ i \in V,\ k \in [K] \tag{A.12}$$

$$\sum_{i\in V} \gamma_{i,k} = |V| - k + 1 \quad \forall\ k \in [K] \tag{A.13}$$

$$\gamma_{i,k}, \delta_{i,k} \in \{0, 1\} \quad \forall\ i \in V,\ k \in [K]. \tag{A.14}$$

For each $k \in [K]$, constraints (A.10) ensure that for every $i \in V$, $\delta_{i,k} = 1 \Rightarrow \alpha_k \leq \omega_i$. Constraints (A.11) ensure that $\delta_{i,k} = 1$ for exactly k elements in $\{\omega_i\}_{i\in V}$. Constraints (A.12) ensure that for $i \in V$, $\gamma_{i,k} = 1 \Rightarrow \alpha_k \geq \omega_i$. Constraints (A.13) ensure that $\gamma_{i,k} = 1$ for exactly $|V| - k + 1$ elements in $\{\omega_i\}_{i\in V}$. Constraints (A.14) define the binary nature of the variables. This linearization is similar to the constraints provided by Rubin, 2015, where the goal is to minimize the sum of the k largest elements in a vector.

The correctness of constraints (A.10)-(A.14) is now shown. It is required to show that a solution satisfies these constraints if and only if it defines α_k for all $k \in [K]$ to be the $k-$th largest element in the set $\{\omega_i\}_{i\in V}$.

For the sufficiency condition, consider a solution $(\alpha_k, \delta_{i,k}, \gamma_{i,k})_{i\in V, k\in[K]}$ that satisfies constraints (A.10)-(A.14). It is required to show that $\alpha_k = \omega_{(k)}$ for all $k \in [K]$. Suppose for the purpose of contradiction that $\alpha_{k'} \neq \omega_{(k')}$ for some $k' \in [K]$. Then, one of the following two cases holds for that k':

- *Case 1:* $\alpha_{k'} < \omega_{(k')}$. Any $i \in V$ with $\gamma_{i,k'} = 1$ must have $i \in S^-_{k'}$, since it must have $\omega_i \leq \alpha_{k'} < \omega_{(k')}$. Hence, $\{i \in V |\ \gamma_{i,k'} = 1\} \subseteq S^-_{k'}$ and it can be shown that $\sum_{i\in V} \gamma_{i,k'} \leq \sum_{i\in S^-_{k'}} 1 = |S^-_{k'}| < |V| - k' + 1$, which provides a contradiction with constraint (A.13).
- *Case 2:* $\alpha_{k'} > \omega_{(k')}$. Any $i \in V$ with $\delta_{i,k'} = 1$ must have $i \in S^+_{k'}$, since it must have $\omega_i \geq \alpha_{k'} > \omega_{(k')}$. Hence, $\{i \in V |\ \delta_{i,k'} = 1\} \subseteq S^+_{k'}$ and it can be shown that $\sum_{i\in V} \delta_{i,k'} \leq \sum_{i\in S^+_{k'}} 1 = |S^+_{k'}| < k'$, which provides a contradiction with constraint (A.11).

Hence, either case contradicts the assumption that $\alpha_{k'} \neq \omega_{(k')}$ for some $k' \in [K]$. Therefore, it can be concluded that $\alpha_k = \omega_{(k)}$ for all $k \in [K]$.

For the necessary condition, for each $k \in [K]$, it is required to show that a solution $(\alpha_k, \delta_{i,k}, \gamma_{i,k})_{i\in V}$ that defines α_k to be the $k-$th largest element in $\{\omega_i\}_{i\in V}$ will satisfy satisfies constraints (A.10)-(A.14). In such a solution, $\alpha_k = \omega_{(k)}$ by definition. The $\delta_{i,k}$ and $\gamma_{i,k}$ values for each $i \in V$ are derived from (A.8) and (A.9). First, consider constraint (A.10). For all $i \in S'(k)$, since $\delta_{i,k} = 1$ and $\alpha_k \leq \omega_i$ by the definitions of $S(k)$ and $S^+(k)$ (whose union is a superset of $S'(k)$), this constraint is satisfied. For all $i \in V \backslash S'(k)$, since $\delta_{i,k} = 0$ and $\alpha_k \leq 1$ by constraint (2.27), this constraint is satisfied. Hence, this solution satisfies constraint (A.10). Further, since

$$\sum_{i\in V} \delta_{i,k} = \sum_{i\in S'(k)} \delta_{i,k} + \sum_{i\in V\backslash S'(k)} \delta_{i,k} \tag{A.15}$$

$$= k + 0 = k, \tag{A.16}$$

this solution satisfies constraint (A.11). Similarly, using the $\{\gamma_{i,k}\}_{i\in V}$ values from (A.9), it can be shown that the solution satisfies constraints (A.12) and (A.13). The proof proceeds similar to the proof for the solution satisfying constraints (A.10) and (A.11), and is hence omitted. Finally, the $\{\delta_{i,k}, \gamma_{i,k}\}_{i\in V}$ values satisfy constraint (A.14) by definition.

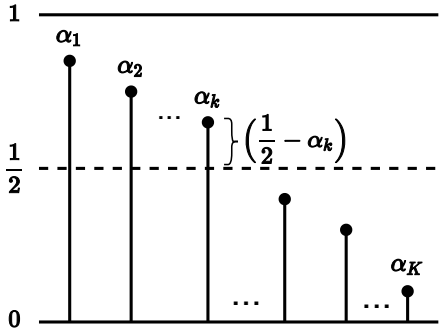


Figure A.1 District votes-shares in K districts sorted in a non-increasing order, $\{\alpha_k\}_{k\in[K]}$.

(c) Third, constraints (2.25) are linearized. Here, for each $k \in [K]$, the value of α_k defined by constraints (A.10)-(A.14) is used to define μ_{km} to be the minimum average vote-share for A to win exactly k districts. To achieve this, the vote-share α_m in each district $m \in [K]$ is decreased by a value of $\alpha_k - \frac{1}{2}$, which is the fraction of voters needed to just flip district k; note that this will increase the vote-share if $\alpha_k < 1/2$. The intuition behind this is illustrated in Figure A.1. Constraints (2.25) are given by,

$$\mu_{km} = \begin{cases} 0, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \leq 0 \\ \alpha_m + \frac{1}{2} - \alpha_k, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \in (0,1) \\ 1, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k \geq 1 \end{cases} \qquad \forall\ k, m \in [K].$$

It can be seen that for each $k, m \in [K]$, μ_{km} is a non-convex piecewise linear function of $(\alpha_m + 1/2 - \alpha_k)$ with 2 breakpoints to ensure that μ_{km} does not leave the $[0,1]$ interval. Linearizing a piecewise linear function in an MIP has been well studied (Grimstad and Knudsen, 2020), and this chapter presents a concise version of the standard big-M constraints. The concision comes from using only 2 binary variables for each $k, m \in [K]$ instead of the standard 3 variables needed when the piecewise linear function has 2 breakpoints. See Vielma et al., 2010 for convex combination models to linearize non-convex piecewise linear functions.

The linearization is now described. For each $k, m \in [K]$, additional binary variables Ω_{km} and Ω'_{km} are defined as follows

$$\Omega_{km} = \begin{cases} 1, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k > 0. \\ 0, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k < 0. \end{cases}$$

$$\Omega'_{km} = \begin{cases} 1, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k < 1. \\ 0, & \text{if } \alpha_m + \frac{1}{2} - \alpha_k > 1. \end{cases}$$

Here, Ω_{km} (Ω'_{km}) arbitrarily takes the value of either 0 or 1 if $\alpha_m + \frac{1}{2} - \alpha_k = 0$ (1). The first idea in the linearization is to use big-M constraints to ensure that Ω_{km} and Ω'_{km} are defined with respect to the value of $\left(\alpha_m + \frac{1}{2} - \alpha_k\right)$. The second idea is to define μ_{km} using Ω_{km}, Ω'_{km} and $\left(\alpha_m + \frac{1}{2} - \alpha_k\right)$. Note that for every $k, m \in [K]$, the expression $\left(\alpha_m + \frac{1}{2} - \alpha_k\right)$ ranges between $-\frac{1}{2}$ and $\frac{3}{2}$. The third claim is that the following set of constraints linearize constraints (2.25). The linearized constraints are given by,

$$\alpha_m + \frac{1}{2} - \alpha_k \leq \frac{3}{2}\Omega_{km} \quad \forall\ k, m \in [K], \tag{A.17}$$

$$\alpha_m + \frac{1}{2} - \alpha_k \geq \mu_{km} + \frac{3}{2}(\Omega_{km} - 1) \quad \forall\ k, m \in [K], \tag{A.18}$$

$$\alpha_m + \frac{1}{2} - \alpha_k \geq -\frac{3}{2}\Omega'_{km} + 1 \quad \forall\ k, m \in [K], \tag{A.19}$$

$$\alpha_m + \frac{1}{2} - \alpha_k \leq \frac{3}{2}(1 - \Omega'_{km}) + \mu_{km} \quad \forall\ k, m \in [K], \tag{A.20}$$

$$-\Omega'_{km} + 1 \leq \mu_{km} \leq \Omega_{km} \quad \forall\ k, m \in [K], \tag{A.21}$$

$$\Omega_{km}, \Omega'_{km} \in \{0,1\} \quad k, m \in [K]. \tag{A.22}$$

For each $k, m \in [K]$, constraint (A.22) establishes the binary domain of both Ω_{km} and Ω'_{km}. To prove that constraints (A.17) - (A.22) linearize constraint (2.25), there are two parts to the proof. First, it is shown that for a given value of $(\alpha_m + \frac{1}{2} - \alpha_k)$, $\mu_{k,m}$ is correctly defined through the variables $\Omega_{k,m}$ and $\Omega'_{k,m}$. Consider the following five cases for $(\alpha_m + \frac{1}{2} - \alpha_k)$:

- *Case 1*: when $\alpha_m + \frac{1}{2} - \alpha_k < 0$, then $\Omega_{km} = 0$ by combining constraints (A.18) and (2.27), and $\Omega'_{km} = 1$ by constraint (A.19). Substituting these values of Ω_{km} and Ω'_{km} into constraint (A.21) gives $\mu_{km} = 0$. These values also satisfy constraints (A.17) and (A.20).
- *Case 2*: when $\alpha_m + \frac{1}{2} - \alpha_k = 0$, then $\Omega'_{km} = 1$ by constraint (A.19). There are two subcases based on the value of Ω_{km}:
  (a) if $\Omega_{km} = 1$, then $\mu_{km} = 0$ by combining constraints (A.18) and (2.27). Constraints (A.17), (A.20), and (A.21) are also satisfied.
  (b) if $\Omega_{km} = 0$, then $\mu_{km} = 0$ by constraint (A.21). Constraints (A.17), (A.18), and (A.20) are also satisfied.
- *Case 3*: when $\alpha_m + \frac{1}{2} - \alpha_k \in (0, 1)$, then $\Omega_{km} = 1$ and $\Omega'_{km} = 1$ by constraints (A.17) and (A.19), respectively. Substituting these values of Ω_{km} and Ω'_{km} into constraints (A.18) and (A.20) gives $\mu_{km} = \alpha_m + \frac{1}{2} - \alpha_k$. These values also satisfy constraint (A.21).
- *Case 4*: when $\alpha_m + \frac{1}{2} - \alpha_k = 1$, then $\Omega_{km} = 1$ by constraint (A.17). There are two subcases based on the value of Ω'_{km}:
  (a) if $\Omega'_{km} = 1$, then $\mu_{km} = 1$ by combining constraints (A.20) and (2.27). Constraints (A.18), (A.19), and (A.21) are also satisfied.
  (b) if $\Omega'_{km} = 0$, then $\mu_{km} = 1$ by constraint (A.21). Constraints (A.18), (A.19), and (A.20) are also satisfied.
- *Case 5*: when $\alpha_m + \frac{1}{2} - \alpha_k > 1$, then $\Omega_{km} = 1$ by constraint (A.17) and $\Omega'_{km} = 0$ by combining constraints (A.20) and (2.27). Substituting these values of Ω_{km} and Ω'_{km} into constraint (A.21) gives $\mu_{km} = 1$. These values also satisfy constraints (A.18) and (A.19).

Therefore, regardless of the value of $\alpha_m + \frac{1}{2} - \alpha_k$, the value of μ_{km} is determined as defined in constraint (2.25).

Second, it is shown that for given values of Ω_{km} and Ω'_{km}, the values of μ_{km} and $(\alpha_m + \frac{1}{2} - \alpha_k)$ are correctly determined. There are 4 cases:

- *Case 1:* $\Omega_{km} = \Omega'_{km} = 0$ is not possible by constraint (A.21).
- *Case 2:* when $\Omega_{km} = 0$ and $\Omega'_{km} = 1$, then $\mu_{km} = 0$ by (A.21), and $(\alpha_m + \frac{1}{2} - \alpha_k) \leq 0$ by constraint (A.17) (Other constraints are also satisfied by noting that $\alpha_m + \frac{1}{2} - \alpha_k \geq -1/2$).
- *Case 3:* when $\Omega_{km} = 1$ and $\Omega'_{km} = 0$, then $\mu_{km} = 1$ by (A.21), and $(\alpha_m + \frac{1}{2} - \alpha_k) \geq 1$ by constraint (A.19) (Other constraints are also satisfied by noting that $\alpha_m + \frac{1}{2} - \alpha_k \leq 3/2$).
- *Case 4:* when $\Omega_{km} = \Omega'_{km} = 1$, then $\mu_{km} = \alpha_m + \frac{1}{2} - \alpha_k$ by combining constraints (A.18) and (A.20). Other constraints are also satisfied.

□

A.5 Adopting the Multilevel Algorithm from Graph Partitioning

The multilevel algorithm presented in Chapter 2 differs from existing multilevel algorithms for graph partitioning in two aspects. First, the objective of a typical graph partitioning problem is to minimize the number of "cut" edges, i.e., edges that cross two partitions. In the coarsening procedure of a typical multilevel

algorithm, a *maximal matching* is found in the current graph level and the matched pair of units (vertices) are merged. Several types of matchings are proposed in the literature, ranging from random matchings to matchings that minimize the sum of edge weights (Karypis and Kumar, 1998). However, for a districting problem where unit populations in the original graph are heterogeneous, the goal of the matching procedure used in this chapter is to coarsen the graph such that the unit populations in the coarse graph are as homogeneous as possible. To achieve this, the edges are given a weight based on the unit populations of the end points and a matching is found such that the maximum edge weight is minimized, as described in Section 2.4.1. This method is essential since it was observed that the presence of highly populous units in the coarse graphs prevents the algorithm from finding district plans that satisfy practical population balance constraints.

Second, solving the inner-level MIP of the districting problem is computationally more challenging than solving a typical graph partitioning problem. This is because of the additional constraints and variables in the districting MIP, such as the flow-based contiguity constraints and the fairness-defining constraints introduced in Section 2.3. Hence, a larger computational resource is needed to solve the inner-level MIPs. This challenge is handled in the presented work by setting large time limits of up to 24 hours for solving each MIP in the optimization stage as discussed in Section 2.5.

A.6 Approximating the Minmax Maximal Matching Problem (MLP)

This section presents an approximation result for the *minmax maximal matching problem* (MLP). Given a connected undirected graph $G = (V, E)$ and edge weights u_e for all $e \in E$, MLP finds a maximal matching M in G that minimizes the maximum edge weight $\max_{e \in M} u_e$. MLP is NP-complete via a reduction from the dominating set problem (Lavrov, 2019). This chapter uses a greedy algorithm to find a feasible solution to MLP; this matching can then be used to merge in the coarsening procedure in Section 2.4.1. For this greedy algorithm, the edge weights for the population-based coarsening is given by $u_{(i,j)} = p_i + p_j$ for every edge $(i, j) \in E$, where $p_i \geq 0$ is the population in every unit $i \in V$.

This section shows that any arbitrary maximal matching (and by extension, the solution found by the greedy algorithm in Section 2.4.1) has an approximation factor that depends on the relative populations of neighboring units in G. To formalize this, let $\rho \coloneqq \max_{(i,j) \in E} \max\{p_i/p_j, p_j/p_i\}$ be the largest ratio of populations among neighboring units in G. Clearly, $\rho \geq 1$. In defining ρ, a key assumption is that unit populations are strictly positive. Theorem 2 shows that the maximum edge weight (u') in any arbitrary maximal matching is at the most ρ times the maximum edge weight (u^*) in an optimal maximal matching to MLP. In the special case when all the unit populations are equal, this result is trivially true since all the edge weights are equal, where $\rho = 1$. In this case, every maximal matching is optimal to MLP. Theorem 2 derives an approximation ratio for the general case for any $\rho \geq 1$.

Theorem 2. *Consider a connected graph $G = (V, E)$, positive unit populations $p_i > 0$ for every unit $i \in V$, and edge weights $u_{ij} = p_i + p_j$ for every edge $(i, j) \in E$. Let M^* be an optimal solution to the minmax maximal matching problem with inputs $(G, \{u_{ij}\}_{(i,j) \in E})$, and let $u^* \coloneqq \max_{e \in M^*} u_e$ be its maximum edge weight. Let M be an arbitrary maximal matching with maximum edge weight $u' \coloneqq \max_{e \in M} u_e$. Then, $u'/\rho \leq u^* \leq u'$, where $\rho \coloneqq \max_{(i,j) \in E} \max\{p_i/p_j, p_j/p_i\}$.*

Proof. Since M is a maximal matching (i.e., M is a feasible solution to the minmax maximal matching problem), $u^* \leq u'$. The proof for $u^* \geq u'/\rho$ proceeds by contradiction. Assume the contrary, that $u^* < u'/\rho$.

Let $e' = (i', j')$ be an edge with maximum weight in M, i.e., $e' \coloneqq \arg\max_{e \in M} u_e$ and $u_{e'} = u'$. There are two cases:

1. *Case 1:* edge e' is a part of M^*. In this case, $u' \leq u^*$ since u^* is the maximum edge weight in M^*. This is contrary to the assumption that $u^* < u'/\rho$ since $\rho \geq 1$.

2. *Case 2:* edge e' is not a part of M^*. Since M^* is a maximal matching, at least one of i' or j' is a part of a matched edge in M^* that is not e'. Without loss of generality, select i' as a unit that is part of a matched edge in M^*. Let that edge be (i', j'') for some $j'' \in V$, and its weight is given by,

$$u_{(i',j'')} = p_{i'} + p_{j''} \geq p_{i'} + \frac{p_{i'}}{\rho} = p_{i'} \frac{(1+\rho)}{\rho}. \tag{A.23}$$

Since (i', j'') is a part of M^*, and u^* is the maximum edge weight in M^*,

$$u_{(i',j'')} \leq u^*. \tag{A.24}$$

From (A.23) and (A.24),

$$p_{i'} \frac{(1+\rho)}{\rho} \leq u^*. \tag{A.25}$$

On the other hand, the weight of edge (i', j') is given by,

$$u' = p_{i'} + p_{j'} \leq p_{i'} + \rho p_{i'} = (1+\rho) p_{i'}. \tag{A.26}$$

From (A.25) and (A.26), it is evident that $u' \leq \rho u^*$. This is contrary to the assumption that $u^* < u'/\rho$. □

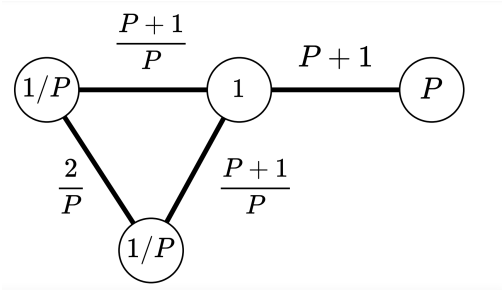


Figure A.2: An example instance of MLP illustrating that Theorem 2 provides a tight approximation; a graph with four units with respective unit populations $1/P$, $1/P$, 1 and P for some $P > 1$. The unit populations and edge weights are depicted at the units and edges, respectively.

It can be shown that the greedy algorithm in Section 2.4.1 provides a tight approximation, i.e, there exists an instance of MLP such that $u'/u^* = \rho$, where u' and u^* are the greedy and optimal solutions, respectively.

Example 1. *Consider an instance of MLP with four units and four edges as depicted in Figure A.2. The respective unit populations are* $1/P$, $1/P$, 1 *and* P *for some* $P > 1$. *The maximum population ratio among neighboring units is* $\rho = P$.

For the instance in Example 1, the optimal maximal matching to MLP selects exactly one edge whose edge weight is $u^* = (P+1)/P$. On the other hand, the greedy algorithm in Section 2.4.1 first sorts the edges in the non-decreasing order: $\{2/P, (P+1)/P, (P+1)/P, P+1\}$. The maximal matching returned by the greedy algorithm selects the first and the fourth edges with respective weights $2/P$ and $P+1$. The maximum edge weight for the greedy solution is $u' = P+1$. Hence, the approximation ratio is $u'/u^* = (P+1)/((P+1)/P) = P$, which is equal to ρ. Note that if the unit populations are restricted to integers, setting P to be an integer and multiplying all the unit populations in this example by P would yield an instance with integral unit populations, and the tightness result still holds.

A.6.1 Initial Solution Heuristic

In the exact optimization stage to solve an MIP, the computational performance of an optimization solver can be improved when the solver is given an initial feasible solution. This section presents a multistart local search heuristic used to generate a feasible solution for a PFDP. A feasible solution is one that satisfies contiguity, population balance, and an $\epsilon-$constraint from (2.35)—(2.37). The heuristic proceeds in two phases. The first phase creates a feasible solution, and the second phase improves the compactness objective using local search.

To generate a feasible solution, the heuristic proceeds as in Bozkaya et al., 2003, Vickrey, 1961, and Thoreson and Liittschwager, 1967, as follows. An arbitrary unit is first picked uniformly at random from V to seed a district, and this district is extended by randomly assigning neighboring units to it. This district is iteratively grown by adding one unassigned neighboring unit at a time. This process continues until either the population in the district exceeds $\overline{P}$ or there are no more unassigned units in any of the neighborhoods of the units in the district. Next, a new unassigned unit is randomly chosen to seed and grow a new district. This proceeds continues until all the units in the graph are assigned to districts. At the end of this procedure, it is possible to have either more than, equal to, or fewer than K districts. If there are more than K districts, pairs of neighboring districts are iteratively merged until the number of districts equals K. Hence, this method finds K contiguous districts. If there are fewer than K districts, or if the solution does not satisfy the population balance or fairness-based $\epsilon-$constraints, it is discarded, and the method is repeated until a feasible solution is found. Based on experiments not included in this chapter, it is observed that the computational time to find an initial solution increases as the feasible region to the districting problem is made smaller, either by tightening the $\epsilon-$constraint or by reducing the population deviation threshold τ.

In addition to finding a feasible solution, it is preferable to find one that is highly compact (i.e., the primary objective). In a local search method, starting with a feasible solution, the neighborhood of the solution is explored for any improvement in the objective (Ricca et al., 2013). The neighborhood considered in this chapter is obtained by re-assigning a single unit from one district to another district to reduce the compactness objective ϕ_{comp}. Note that such a re-assignment is referred to as a "flip" operation in algorithms that generate random district plans using Markov chains, e.g., DeFord et al., 2021a. After every such re-assignment, the feasibility of the solution is assessed. The solution in the neighborhood with the smallest compactness value, ϕ_{comp}, is chosen as the starting point for the next iteration, and the local search terminates either when the number of iterations reaches a limit, or when there is no further improvement. The quality of the best solution from the local search method is typically influenced by the initial solution. To encourage a more complete exploration of the solution space, a multistart local search procedure is used, where multiple feasible solutions are generated as starting points for local search. The initial solution for the MIP is the solution with the smallest ϕ_{comp} produced by the multistart local search.

A.7 Computational Results for Pareto-Frontier

A.7.1 Efficiency Gap and Compactness

Table A.1: Computational results from the $\epsilon-$constraint method to obtain the Pareto-frontier between the efficiency gap and compactness, when coarsening using maximum matchings (MM) and maximal matchings (ML). An optimality gap of "-" indicates that the solution is optimal, i.e., the upper bound (UB) and lower bound (LB) on the compactness objective found in the branch-and-cut algorithm are equal.

Coarsening	ϕ_{EG}	ϕ_{comp} ($\times 10^9$)	Clock time (s)	CPU time (ticks)	# B&C nodes	Opt. gap ($\frac{UB-LB}{UB}$)
ML	0.1561	2.683	416	180,309	112	-
	0.1511	2.683	784	290,876	199	-
	0.0277	2.686	1,161	467,881	987	-
	0.0272	2.688	28,927	9,402,030	7,091	-
	0.0257	2.688	8,480	3,010,334	3,187	-
	0.0234	2.689	1,695	638,511	1,756	-
	0.0224	2.690	1,625	599,558	2,347	-
	0.0220	2.691	12,849	4,160,289	4,860	-
	0.0217	2.695	81,616	24,191,404	16,204	-
	0.0215	2.695	52,968	15,764,469	12,876	-
	0.0198	2.697	13,370	3,801,803	10,131	-
MM	0.0307	3.923	295	134,708	123	-
	0.0290	3.932	7,584	2,452,971	4,200	-
	0.0286	3.937	10,582	3,517,502	5,491	-
	0.0283	3.943	9,689	2,610,137	3,907	-
	0.0281	3.952	23,973	7,019,182	14,997	-
	0.0274	3.954	15,944	5,100,907	7,839	-
	0.0268	3.961	14,622	4,349,295	8,056	-
	0.0255	3.966	3,289	1,093,726	6,296	-
	0.0252	3.973	45,793	14,142,057	23,586	-
	0.0251	3.979	32,342	9,499,810	16,455	-
	0.0250	3.982	5,678	1,828,732	11,261	-
	0.0243	3.983	48,467	15,287,890	32,957	-
	0.0235	3.989	34,041	10,235,375	22,377	-
	0.0190	4.003	39,687	12,684,359	44,313	-

A.7.2 Partisan Asymmetry and Compactness

Table A.2: Computational results from the $\epsilon-$constraint method to obtain the Pareto-frontier between the partisan asymmetry and compactness objectives, when coarsening using maximum matchings (MM) and maximal matchings (ML). *The solver takes extra time beyond the 6 hour MIP time limit to completely terminate.

Coarsening	ϕ_{PA}	$\phi_{comp}(\times 10^9)$	Clock time (s)	CPU time (ticks)	# B&C nodes	Opt. gap ($\frac{UB-LB}{UB}$)
ML	0.0633	3.166	21,601*	5,457,399	4,285	0.154
	0.0127	3.277	21,600	6,187,473	1,554	0.184
	0.0020	3.535	21,600	6,141,634	6,796	0.243
	0.0016	3.929	21,600	5,934,803	3,484	0.319
MM	0.0034	4.605	21,608*	6,102,360	14,573	0.148
	0.0024	5.516	21,600	6,034,105	14,503	0.289
	0.0018	5.800	21,600	6,045,607	8,643	0.324
	0.0002	6.729	21,600	5,861,623	4,098	0.420

A.7.3 Competitiveness and Compactness

Table A.3: Computational results from the $\epsilon-$constraint method to obtain the Pareto-frontier between competitiveness and compactness objectives, when coarsening using maximum matchings (MM) and maximal matchings (ML). From the 72 solutions obtained when coarsening using ML, a subset of 18 solutions are showcased here for simplicity. This subset is selected by picking every fourth solution in the sorting of the 72 solutions according to their respective competitiveness (ϕ_{cmpttv}) values. An optimality gap of "-" indicates that the solution is optimal, i.e., the upper bound (UB) and lower bound (LB) on the compactness objective found in the branch-and-cut algorithm are equal.

Coarsening	ϕ_{cmpptv}	$\phi_{comp}(\times 10^9)$	Clock time (s)	CPU time (ticks)	# B&C nodes	Opt. gap ($\frac{UB-LB}{UB}$)
ML	0.333	2.691	897	443,614	642	-
	0.313	2.699	590	251,040	1,208	-
	0.303	2.709	651	296,992	1,211	-
	0.290	2.714	475	220,096	426	-
	0.276	2.724	574	278,495	1,608	-
	0.266	2.738	529	243,615	807	-
	0.258	2.746	520	239,000	604	-
	0.248	2.761	693	317,177	2,105	-
	0.239	2.768	755	349,210	2,952	-
	0.231	2.781	825	383,338	2,746	-
	0.223	2.788	972	496,707	548	-
	0.213	2.802	1,639	869,083	1,916	-
	0.163	2.875	1,755	809,647	4,702	-
	0.153	2.895	11,345	3,633,036	7,867	-
	0.138	2.905	1,825	710,101	1,506	-
	0.123	2.923	11,290	3,416,250	2,539	-
	0.111	2.939	16,952	5,092,444	3,451	-
	0.099	2.959	33,652	11,747,083	5,430	-
MM	0.286	3.923	1,094	503,219	918	-
	0.265	3.935	321	156,208	182	-
	0.247	3.965	2,866	1,029,225	3,429	-
	0.246	3.972	11,033	3,425,895	5,827	-
	0.238	3.988	995	417,346	1,199	-
	0.234	3.994	444	203,912	414	-
	0.224	4.018	1,025	463,780	2,001	-
	0.188	4.081	4,212	1,368,974	2,730	-
	0.186	4.088	6,886	2,225,135	4,722	-
	0.171	4.089	3,115	1,095,628	3,175	-
	0.159	4.103	556	253,614	616	-
	0.135	4.131	16,198	5,681,976	9,785	-
	0.131	4.146	2,928	1,161,383	3,175	-
	0.129	4.154	5,835	2,082,055	6,933	-
	0.125	4.155	3,121	1,083,149	5,181	-
	0.121	4.157	2,255	780,248	4,666	-
	0.119	4.164	2,952	1,000,392	4,791	-

Appendix B

Appendix for Chapter 3

B.1 Pseudocodes

This appendix presents pseudocodes for the components of the optimization framework in Algorithm 3.1. Appendix B.1.1 details pseudocodes for the local search subroutines applied in stages 1 through 4. Appendix B.1.2 introduces the pseudocode for stage 1 of the algorithm, which generates an initial district map. Stages 2 through 4 are direct calls to the local search subroutines in Appendix B.1.1.

B.1.1 Local Search Subroutines

This section provides pseudocodes for the local search subroutines implemented in stages 1 through 4. We consider Flip-based and Recom-based neighborhoods. Recall that Flip reassigns units one at a time, whereas Recom merges two districts and reassigns the units by drawing a new boundary between them. We provide two pseudocodes based on the termination criteria and whether non-improving solutions are accepted. The first type, detailed in Algorithm B.1, accepts a local search reassignment only if it improves or maintains the best objective value, and it terminates after N_{iter} consecutive iterations without improvement. We use this approach to minimize $-\phi_{\text{VRA}}$ in stage 1, ϕ_{compact} in stage 2, $-\phi_{\text{cmpttv}}$ in stage 3, and ϕ_{bal} in stage 4. The second type, used in improving ϕ_{bal} in stage 1, allows a local search reassignment only if it strictly improves ϕ_{bal}, and terminates when no further improvements in the objective are possible after considering all potential local reassignments. Algorithm B.2 details this approach.

The reasoning for these methodological variations is as follows. When improving population balance within Stage 1, the goal is to rapidly achieve a 1%-balanced district map; if this is not possible, stage 1 restarts from scratch. Preliminary investigations indicated that when the algorithm accepts non-improving reassignments (as in Algorithm B.1), it can get trapped in a cycle of reassigning zero-population units repetitively. This often results in the algorithm terminating prematurely without achieving a 1%-balanced map. To mitigate this issue, Algorithm B.2 is designed to exclusively accept reassignments that enhance population balance in stage 1. This modification yielded faster termination of this subroutine, which in turn expedites the execution of stage 1 of the algorithm.

Algorithm B.1 iteratively performs Flip-based or Recom-based local search to improve a given map z for a chosen objective ϕ, while ensuring every map produced satisfies a given set of constraints. Lines 1 and 2 initiate the current best district map and its objective. Lines 4-16 iterate until the objective ϕ does not improve in N_{iter} consecutive iterations. For Flip-based local search, lines 5-7 reassign a random *boundary*

Algorithm B.1: LocalSearch-FixedIterations (G, z, ϕ, N_{iter}, `method`, `constraints`)

Inputs : Graph G, district map z, minimization objective ϕ, `constraints` $\subseteq$ {contiguity, 1%-population balance, VRA districts, maximum compactness limit, minimum competitiveness limit}

Parameters : Maximum number of consecutive local search iterations with no improvement $N_{\text{iter}} \in \mathbb{Z}^+$, `method` = {Flip, Recom}

Output : District map z_{best}

1 $\phi^* \leftarrow \phi(z)$
2 $z_{\text{best}} \leftarrow z$
3 **while** ϕ *improves at least once in the last* N_{iter} *iterations* **do**
4 **if** `method` = Flip **then**
5 $u \leftarrow$ Select a random boundary unit in z_{best}
6 $k' \leftarrow$ Select a random district neighboring u in z_{best}
7 $z' \leftarrow$ Reassign unit u to district k' in z_{best}
8 **else if** `method` = Recom **then**
9 $k, k' \leftarrow$ Select a random pair of neighboring districts
10 $G[V_k \cup V_{k'}] \leftarrow$ Subgraph induced by merging districts k and k'
11 $T, T' \leftarrow$ Balanced bisection of a random spanning tree in $G[V_k \cup V_{k'}]$
12 $z' \leftarrow$ Reassign $V_k = T$ and $V_{k'} = T'$ in z_{best}
13 **if** z' satisfies `constraints` **then**
14 **if** $\phi(z') \leq \phi^*$ **then**
15 $z_{\text{best}} \leftarrow z'$
16 $\phi^* \leftarrow \phi(z')$
17 **return** z_{best}

unit (i.e., a unit with a neighbor belonging to a different district) u to a random neighboring district k'. For Recom-based local search, lines 9-12 merge two randomly chosen districts, and find a balanced bisection of a random spanning tree in the combined subgraph using DeFord et al., 2021a. The modified district map is accepted only if it satisfies the given set of constraints and its objective is not worse than the current best. The method terminates with the best map found within the iteration limits.

Algorithm B.2 is a Flip-based local search method used to improve population balance ϕ_{bal} in stage 1 of our algorithm. Lines 1 and 2 initiate the current best map and best objective. Each iteration in lines 4-13 considers all the possible unit reassignments. When a particular reassignment strictly improves ϕ_{bal}, it is accepted, and the loop continues. If no potential reassignment improves ϕ_{bal}, the method terminates with a locally optimal district map.

B.1.2 Stage 1: Create an Initial District Map

Algorithm B.3 produces an initial district map (in stage 1) that meets three criteria: contiguity, a 1%-balance, and the required number of VRA districts. Line 1 generates a contiguous district map using a multi-kernel growth method from Vickrey, 1961. Line 2 generates a locally optimal district map with respect to the population balance objective ϕ_{bal} using Algorithm B.2. If the population deviation of this map is within the 1%-threshold, the algorithm proceeds; otherwise line 1 is restarted. Line 5 evaluates the number of majority-minority districts. If it is less than N_{VRA}, lines 7-11 iteratively construct VRA districts one at a time. Each iteration uses Algorithm B.1 to maximize the VRA score ϕ_{VRA} defined in Equation (3.4). Note that Algorithm B.1 terminates upon reaching the iteration limit, or when ϕ_{VRA} crosses the α_{VRA} threshold needed for a district to be majority-minority. If the former case occurs in any iteration, the algorithm restarts

Algorithm B.2: LocalSearch-UntilNoImprovement (G, z)

Inputs : Graph G, district map z
Parameters: None
Output : District map z_{best}

$\phi^* \leftarrow \phi_{\text{bal}}(z)$
$z_{\text{best}} \leftarrow z$
while ***True*** **do**
  $\mathcal{T} \leftarrow$ Randomly ordered set of potential reassignments in z_{best}
  `// each element` (u, k') `in` $\mathcal{T}$ `is a boundary unit` u `and a district` k' `neighboring` u
  for $(u, k') \in \mathcal{T}$ **do**
    $z' \leftarrow$ Reassign unit u to district k' in z_{best}
    if z' is contiguous **then**
      if $\phi_{\text{bal}}(z') < \phi^*$ **then**
        $z_{\text{best}} \leftarrow z'$
        $\phi^* \leftarrow \phi_{\text{bal}}(z')$
        Break
  if no potential reassignment in $\mathcal{T}$ improves ϕ_{bal} **then**
    Break
return z_{best}

Algorithm B.3: Stage 1: Create an initial district map

Inputs : Coarse graph G_{coarse}, number of districts $K \in \mathbb{Z}^+$, requisite number of majority-minority districts $N_{\text{VRA}} = 1, 2, \ldots, K$, objectives ϕ_{bal} and ϕ_{VRA}, minority fractional threshold $\alpha_{\text{VRA}} \in [0, 1]$
Parameters: Maximum number of consecutive local search iterations with no improvement $N^1_{\text{iter}} \in \mathbb{Z}^+$ for stage 1
Output : District map z_{initial}

1 $z_{\text{contig}} \leftarrow$ Multi-kernel growth from Vickrey, 1961
2 $z_{\text{balanced}} \leftarrow$ LocalSearch-UntilNoImprovement (G_{coarse}, z_{contig})
3 **if** $\phi_{\text{bal}}(z_{\text{balanced}}) > 0.01$ **then**
4 Restart Line 1
5 $Q \leftarrow$ Majority-minority districts in z_{balanced}
6 $z_{\text{VRA}} \leftarrow z_{\text{balanced}}$
7 **while** $|Q| < N_{\text{VRA}}$ **do**
8 $z_{\text{VRA}} \leftarrow$ LocalSearch-FixedIterations (G_{coarse}, z_{VRA}, $-\phi_{\text{VRA}}$, N^1_{iter}, Recom, {contiguity, 1%-balance})
9 **if** Line 8 terminated without a new VRA district **then**
10 Restart Line 1
11 $Q \leftarrow$ Majority-minority districts in z_{VRA}
12 $z_{\text{initial}} \leftarrow z_{\text{VRA}}$
13 **return** z_{initial}

from scratch. Otherwise, when the requisite number of VRA districts are created, this method terminates with an initial district map.

B.2 Map A With Three Competitive Districts

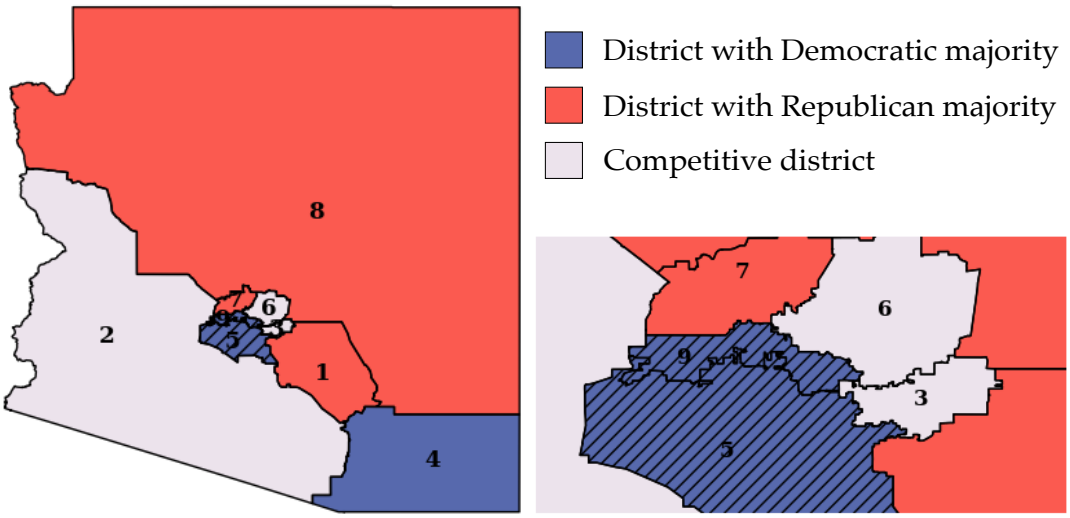


Figure B.1: District map A; all nine districts (left) and districts in the city of Phoenix (right). The two majority-Hispanic districts are shaded.

Table B.1: The demographics for the nine districts in district map A. *The number of elections among nine past elections (2018 President and U.S. Senate; 2020 U.S. Senate, Governor, Secretary of State, Attorney General, State Treasurer, Superintendent of Public Instruction and State Mine Inspector) that each party has a majority in each district. **The party that has a simple majority in the average of the nine elections.

District	Total Population		Racial Demographics			Competitiveness			
	Population	Population Deviation	Hispanic (%)	NH White (%)	Other Minority %	Vote Spread %	D Wins*	R Wins*	Party with a simple majority**
1	794,612	1	22.29	65.91	11.8	17.15	0	9	R
2	794,610	1	45.59	45.84	8.57	5.8	0	9	R
3	794,611	0	24.91	63.66	11.43	5.5	1	8	R
4	794,610	1	36.03	53.85	10.12	13.15	9	0	D
5	794,612	1	51.86	30.98	17.16	26.47	9	0	D
6	794,612	1	18.7	69.53	11.77	3.8	4	5	R
7	794,612	1	19.38	70.62	9.99	18.33	0	9	R
8	794,611	0	14.55	61.69	23.76	14.88	0	9	R
9	794,612	1	51.49	33.89	14.62	26.77	9	0	D

B.3 Map B With Seven Competitive Districts

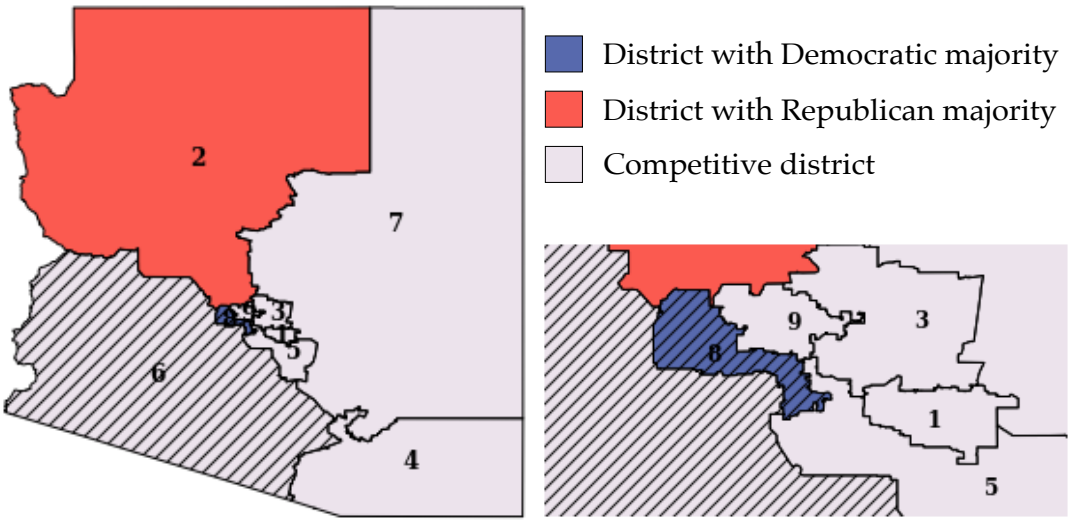


Figure B.2: District map B; all nine districts (left) and districts in the city of Phoenix (right). The two majority-Hispanic districts are shaded.

Table B.2: The demographics for the nine districts in district map B. *The number of elections among nine past elections (2018 President and U.S. Senate; 2020 U.S. Senate, Governor, Secretary of State, Attorney General, State Treasurer, Superintendent of Public Instruction and State Mine Inspector) that each party has a majority in each district. **The party that has a simple majority in the average of the nine elections.

District	Total Population		Racial Demographics			Competitiveness			
	Population	Population Deviation	Hispanic (%)	NH White (%)	Other Minority %	Vote Spread %	D Wins*	R Wins*	Party with a simple majority**
1	794,610	1	26.06	62.19	11.74	5.72	1	8	R
2	794,613	2	14.72	73.72	11.56	25.28	0	9	R
3	794,609	2	18.7	70.13	11.16	5.77	0	9	R
4	794,612	1	44.93	46.15	8.93	6.96	8	1	D
5	794,613	2	26.17	57.4	16.43	2.82	7	2	D
6	794,611	0	51.19	36.3	12.51	0.57	7	2	D
7	794,614	3	20.96	57.41	21.64	5.13	8	1	D
8	794,612	1	52.03	35.22	12.75	13.28	8	1	D
9	794,608	3	29.83	57.42	12.75	1.56	5	4	R

B.4 Map C With Eight Competitive Districts

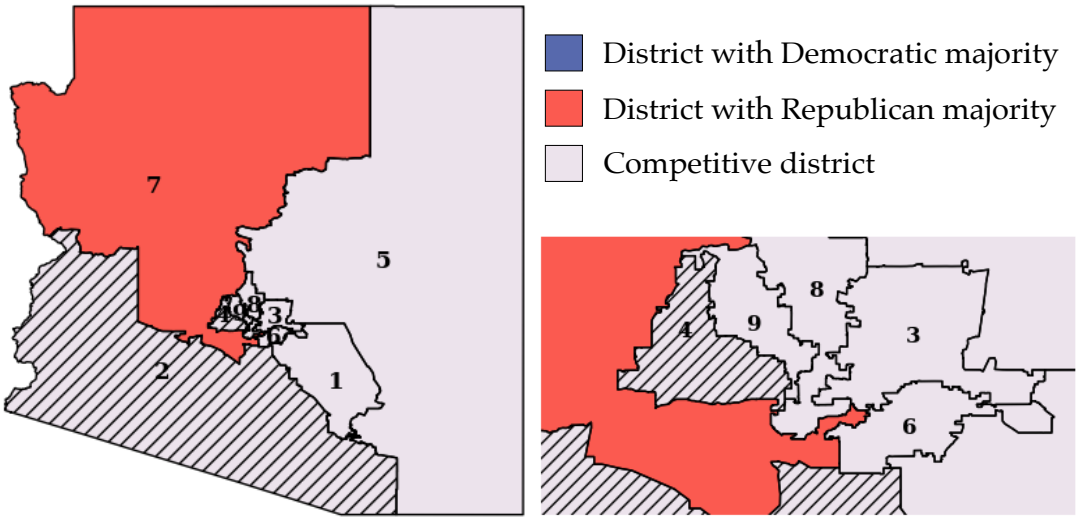


Figure B.3: District map C; all nine districts (left) and districts in the city of Phoenix (right). The two majority-Hispanic districts are shaded.

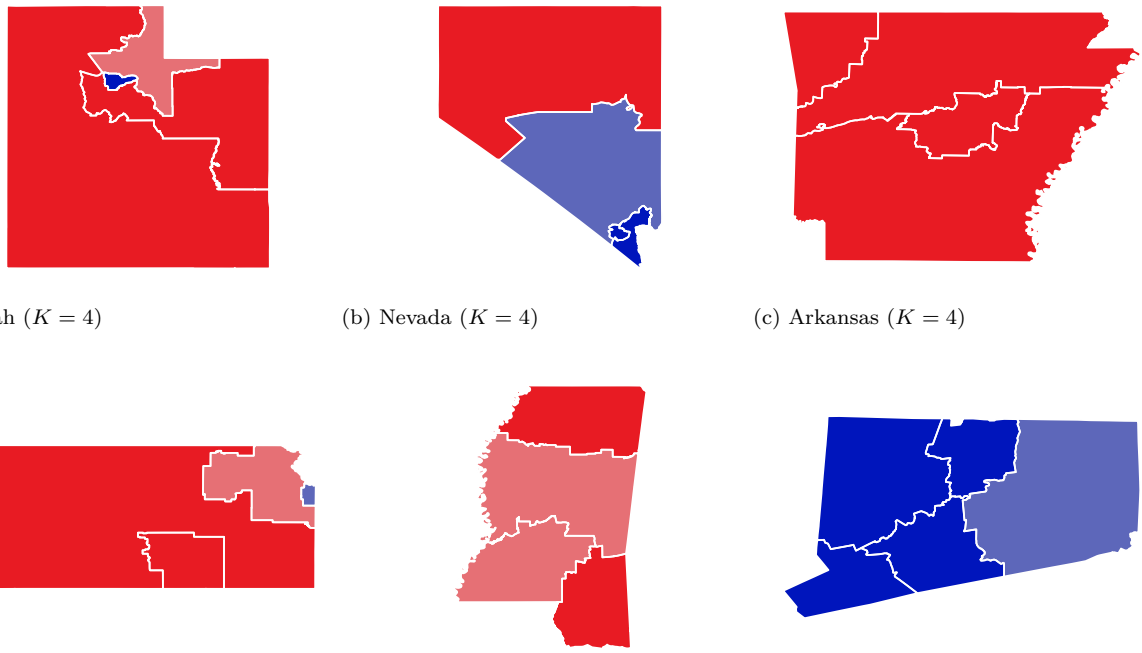
Table B.3: The demographics for the nine districts in district map C. *The number of elections among nine past elections (2018 President and U.S. Senate; 2020 U.S. Senate, Governor, Secretary of State, Attorney General, State Treasurer, Superintendent of Public Instruction and State Mine Inspector) that each party has a majority in each district. **The party that has a simple majority in the average of the nine elections.

District	Total Population		Racial Demographics			Competitiveness			
	Population	Population Deviation	Hispanic (%)	NH White (%)	Other Minority %	Vote Spread %	D Wins*	R Wins*	Party with a simple majority**
1	794,613	2	26.16	62.76	11.08	3.67	1	8	R
2	794,614	3	52.78	38.04	9.18	4.23	7	2	D
3	794,612	1	19.52	67.58	12.9	3.41	4	5	R
4	794,612	1	51.92	36.29	11.79	6.95	8	1	D
5	794,609	2	22.37	55.85	21.78	0.95	5	4	R
6	794,610	1	25.55	60.73	13.72	3.07	4	5	R
7	794,610	1	19.92	66.71	13.36	18.07	0	9	R
8	794,613	2	30.89	56.4	12.72	5.21	7	2	D
9	794,609	2	35.56	51.61	12.83	0.44	5	4	D

Appendix C

Appendix for Chapter 4

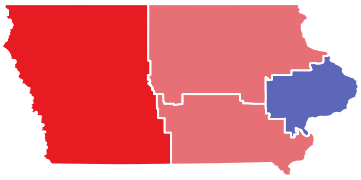
C.1 District Plans Obtained From Applying the Bisection Heuristic to 18 States



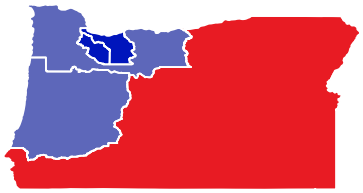
(a) Utah ($K = 4$) (b) Nevada ($K = 4$) (c) Arkansas ($K = 4$)

(d) Kansas ($K = 4$) (e) Mississippi ($K = 4$) (f) Connecticut ($K = 5$)

Figure C.1 District plans obtained from the bisection protocol when D draws the first round.

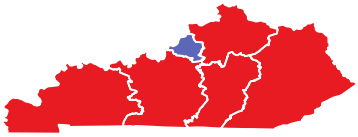


(a) Iowa ($K = 4$)

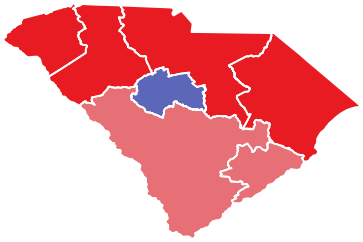


(b) Oregon ($K = 5$)

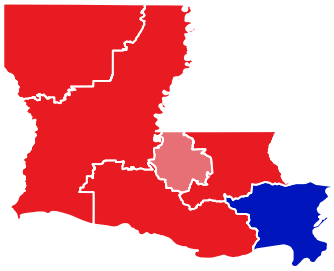
(c) Oklahoma ($K = 5$)



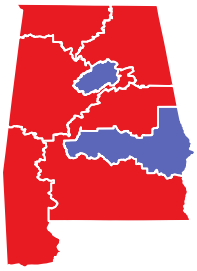
(d) Kentucky ($K = 6$)



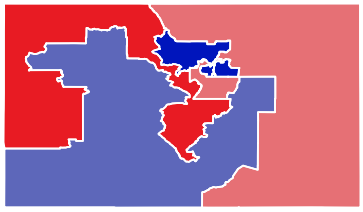
(e) South Carolina ($K = 7$)



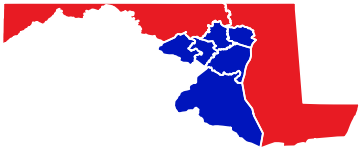
(f) Louisiana ($K = 7$)



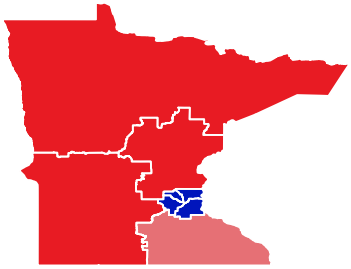
(g) Alabama ($K = 7$)



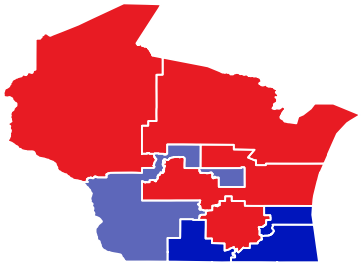
(h) Colorado ($K = 7$)



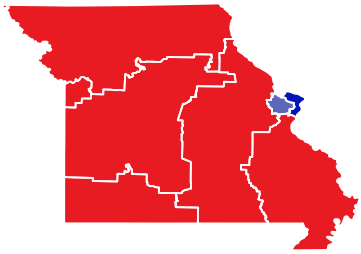
(i) Maryland ($K = 8$)



(j) Minnesota ($K = 8$)



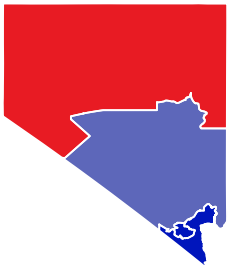
(k) Wisconsin ($K = 8$)



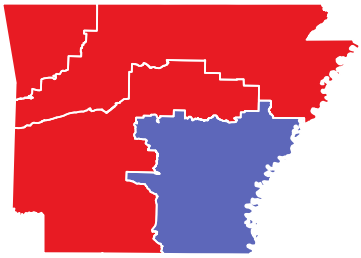
(l) Missouri ($K = 8$)

Figure C.1 (Continued) District plans obtained from the bisection protocol when D draws the first round.

(a) Utah ($K = 4$)

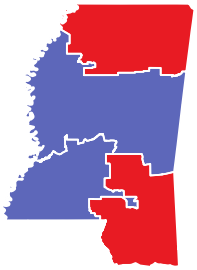


(b) Nevada ($K = 4$)

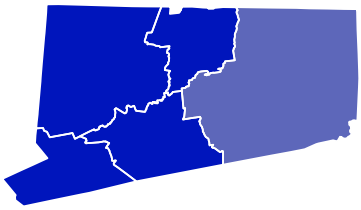


(c) Arkansas ($K = 4$)

(d) Kansas ($K = 4$)

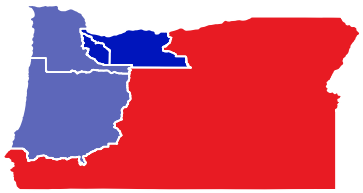


(e) Mississippi ($K = 4$)



(f) Connecticut ($K = 5$)

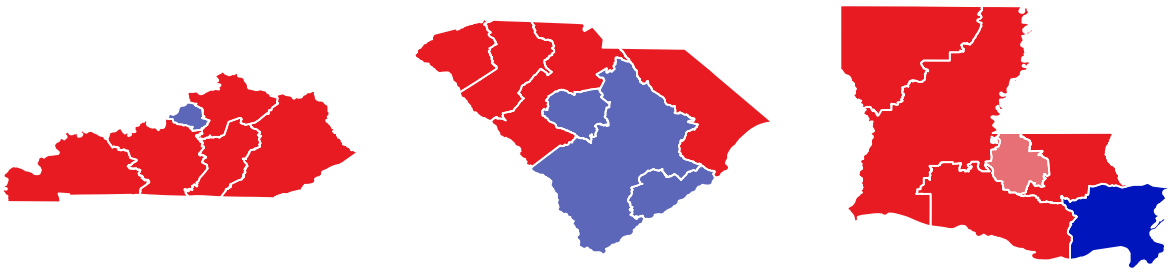
(g) Iowa ($K = 4$)



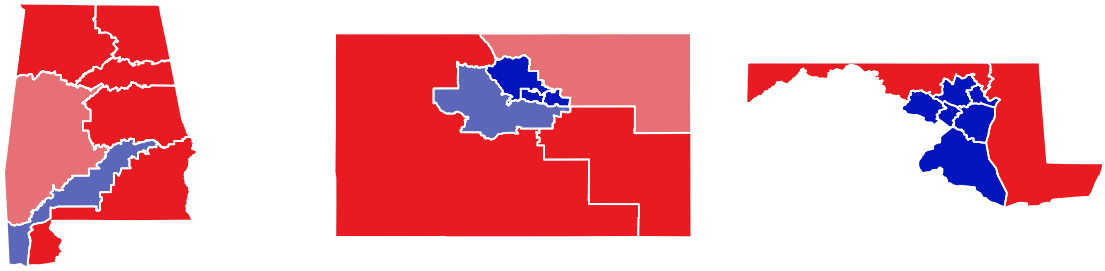
(h) Oregon ($K = 5$)

(i) Oklahoma ($K = 5$)

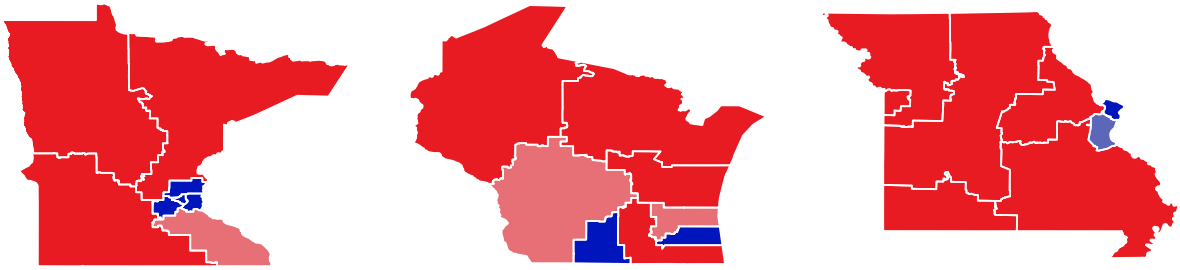
Figure C.2 District plans obtained from the bisection protocol when R draws the first round.



(a) Kentucky ($K = 6$) (b) South Carolina ($K = 7$) (c) Louisiana ($K = 7$)



(d) Alabama ($K = 7$) (e) Colorado ($K = 7$) (f) Maryland ($K = 8$)

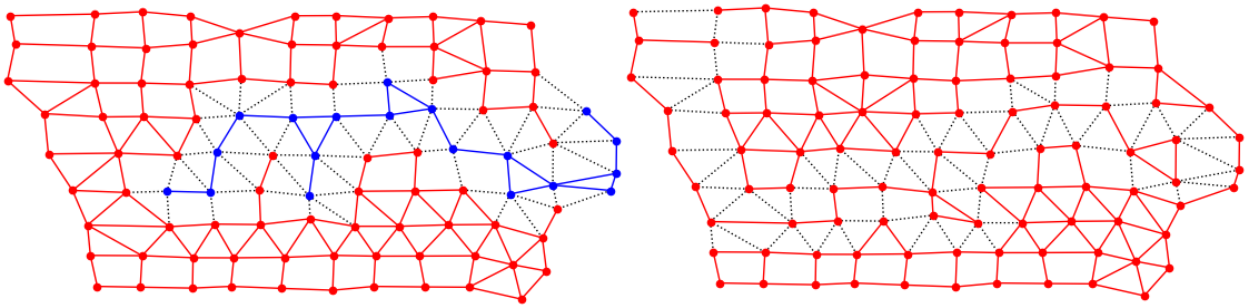


(g) Minnesota ($K = 8$) (h) Wisconsin ($K = 8$) (i) Missouri ($K = 8$)

Figure C.2 (Continued) District plans obtained from the bisection protocol when R draws the first round.

C.2 Motivation for a Compactness Objective in the Bisection Heuristic

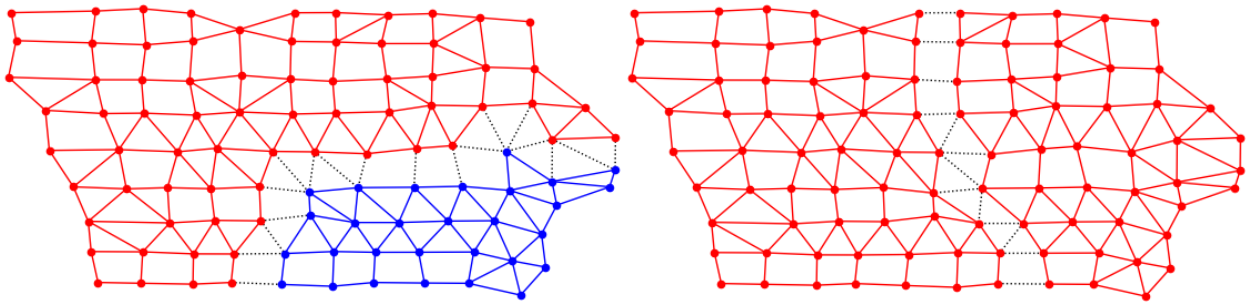
This section motivates the inclusion of a compactness objective in the bisection heuristic (Section 4.3.1) for the discrete geometric (DG) setting. The heuristic's performance with and without a compactness objective is compared using an illustrative case in Iowa's districting. Recall that in the DG setting, every round of bisection divides a given region into two connected and balanced pieces. Following the vote-share allocation in the continuous non-geometric (CN) setting, the bisection heuristic proposes that the drawing player allocates the same amount of vote-share as in the CN setting. Bisecting each round with these vote-share requirements amounts to solving a feasibility problem modeled by MIP constraints (4.1b)-(4.1q). If the problem is infeasible, the vote-share requirements are reduced, and the MIP is solved again; this is repeated until a feasible solution is found.



(a) D draws the first round

(b) R draws the first round

Figure C.3: Bisection heuristic applied to Iowa's county-level instance *without* a compactness objective; panels (a) and (b) depict the first round bisection in G when (a) D and (b) R draws the first round. Solid edges are edges within each of the two pieces, while dotted edges are cut-edges.



(a) D draws the first round

(b) R draws the first round

Figure C.4: Bisection heuristic applied to Iowa's county-level instance *with* a compactness objective; panels (a) and (b) depict the first round bisection in G when (a) D and (b) R draws the first round. Solid edges are edges within each of the two pieces, while dotted edges are cut-edges.

For some instances, solving the feasibility problem results in a bisection with pieces with "non-compact" shapes. This is a problem because a non-compact bisection in the first round could drastically reduce the number of feasible bisections in subsequent rounds. To illustrate this, consider the first round of bisecting the Iowa instance (Section 4.4); this instance requires two rounds of bisection. The shapes of the bisections produced when either player draws first (depicted in Figure C.3) are "snake-like". The second round, regardless of the first player, yields no feasible bisection in either of the two pieces. This demonstrates how the first round bisection may prevent the game from generating any feasible solutions.

Consider the bisection MIP with a compactness objective (4.1a) that minimizes the number of cut-edges (edges that cross two pieces) in the bisection. When the objective is included, the Iowa instance yields a "compact" first round bisection (depicted in Figure C.4). The second round, regardless of the first player, yields a feasible bisection in both pieces. See Section 4.4 for an analysis on the second round bisection. Hence, the inclusion of the compactness-objective is necessary for the bisection heuristic to produce a feasible district plan.

Appendix D

Appendix for Chapter 5

D.1 Proofs for Theoretical Results

This appendix provides proofs for the theoretical results from Section 5.4.

D.1.1 Proof for Theorem 3

Theorem 3. *(Characterization of a Q-proper partition) Given a graph G, positive integers $K \geq 2$ and $Q \geq 1$, a K-partition $\mathcal{P} = \{V^{(k)}\}_{k\in[K]}$ is Q-proper if and only if for every $k \in [K]$,*

(i) $|V^{(k)}| \geq Q + 1$, and

(ii) for every distinct $a, b \in V^{(k)}$, $|C \cap V^{(k)}| \geq Q$ for every a, b-separator $C \in \mathcal{C}_{a,b,G}$.

Proof: ($\Rightarrow$) Consider any part $k \in [K]$. To show the first condition, we note that since $\mathcal{P}$ is Q-proper, $G[V^{(k)}]$ is Q-connected and therefore $|V^{(k)}| \geq Q + 1$ by definition. To show the second condition, we first consider the case where part k forms a complete graph; then $\mathcal{C}_{a,b,G} = \emptyset$ for every $a, b \in V^{(k)}$, and so the second condition holds by default since there are no separators to consider. If part k is not a complete graph, then we first claim that for every $k \in [K]$, every distinct $a, b \in V^{(k)}$ and $C \in \mathcal{C}_{a,b,G}$, $C \cap V^{(k)}$ is an a, b-separator in $G[V^{(k)}]$. We proceed using proof by contradiction; assume the contrary that for some $k \in [K]$, $a, b \in V^{(k)}$ and $C \in \mathcal{C}_{a,b,G}$, $C \cap V^{(k)}$ is not an a, b-separator in $G[V^{(k)}]$. Then, there exists an a, b-path P in $G[V^{(k)}] - C$. Since $V^{(k)} \subseteq V$, path P is also in $G - C$. This is a contradiction since C is an a, b-separator in G. Hence, $C \cap V^{(k)}$ is an a, b-separator in $G[V^{(k)}]$. Therefore, $|C \cap V^{(k)}| \geq Q$ since $G[V^{(k)}]$ is Q-connected.

($\Leftarrow$) We are given that for every $k \in [K]$, (i) $|V^{(k)}| \geq Q + 1$, and (ii) for every distinct $a, b \in V^{(k)}$ and $C \in \mathcal{C}_{a,b,G}$, $|C \cap V^{(k)}| \geq Q$. We need to show that every part is Q-connected. If a part $k \in [K]$ forms a complete graph, $\kappa(G[V^{(k)}]) \geq Q$ since it contains at least $Q + 1$ vertices, and is therefore Q-connected. If a part $k \in [K]$ is not a complete graph, we proceed using proof by contradiction; assume the contrary that $G[V^{(k)}]$ is not Q-connected, i.e., there exists a cutset $S \subset V^{(k)}$ with $|S| = Q' < Q$ that disconnects $G[V^{(k)}]$ into two parts, say A and B. Select arbitrary vertices $a \in A$ and $b \in B$. By construction, S is an a, b-separator in $G[V^{(k)}]$, but S may not be an a, b-separator in G. Define $W = V \backslash V^{(k)}$, and $C = S \cup W$. Any a, b-path P in G must pass through C (i.e., since either P lies entirely in $V^{(k)}$ and must include a vertex from $S \subset C$, or P leaves $V^{(k)}$ and must include a vertex from $W \subset C$), so $C \in \mathcal{C}_{a,b,G}$. By construction, $|C \cap V^{(k)}| = |S| < Q$, which is a contradiction. □

D.1.2 Proof for Corollary 1

Corollary 1. *Given a graph G and $K \geq 2$, consider a solution to the Hess model $\hat{\boldsymbol{x}} \in \mathcal{P}_{HESS}$. For $Q \geq 1$, $\hat{\boldsymbol{x}}$ is Q-proper if and only if $\hat{\boldsymbol{x}}$ satisfies constraints (5.6b)-(5.6c).*

Proof: This proof uses Theorem 3. Given $\hat{\mathbf{x}}$, we first construct the corresponding K-partition $\mathcal{P} = \{V^{(k)}\}_{k\in[K]}$ as follows. Recall from Section 5.3.3 that $R(\hat{\mathbf{x}})$ denotes the set of roots in $\hat{\mathbf{x}}$, and V_r denotes the set of vertices assigned to root $r \in R(\hat{\mathbf{x}})$. Since $\hat{\mathbf{x}}$ satisfies constraints (5.2c), there are exactly K roots. Each root $r \in R(\hat{\mathbf{x}})$ is labeled with a part $k \in [K]$ and set $V^{(k)} = V_r$. From constraints (5.2d) and (5.2e), we can see that $\cup_{k\in[K]} V^{(k)} = V$ and the parts are mutually exclusive. Hence, $\mathcal{P} = \{V^{(k)}\}_{k\in[K]}$ is the K-partition of G that corresponds to the solution $\hat{\mathbf{x}}$. Note that $\mathcal{P}$ can be equivalently written as $\{V_r\}_{r\in\mathcal{R}(\hat{\mathbf{x}})}$. We now prove both directions of the theorem.

($\Rightarrow$) We are given that $\hat{\mathbf{x}}$ is Q-proper. Hence, its representation $\mathcal{P}$ is Q-proper and Theorem 3 implies that for every $r \in \mathcal{R}(\hat{\mathbf{x}})$, (i) $|V_r| \geq Q+1$, and (ii) for every distinct $a, b \in V_r$, $|C \cap V_r| \geq Q$ for every a,b-separator $C \in \mathcal{C}_{a,b,G}$. To show that $\hat{\mathbf{x}}$ satisfies constraints (5.6b), consider each $r \in V$, $a \in V$, $b \in V\backslash\{a\}$, $C \in \mathcal{C}_{a,b,G}$, and the corresponding constraint is $\sum_{c\in C} \hat{x}_{rc} \geq Q\hat{x}_{ra}\hat{x}_{rb}$. Note that this constraint does not exist for cases where $\mathcal{C}_{a,b,G} = \emptyset$ (i.e., a and b are neighbors in G). There are two cases: (i) $r \in \mathcal{R}(\hat{\mathbf{x}})$ and $a, b \in V_r$, in which case constraint (5.6b) can be written as $|C \cap V_r| \geq Q$, which is satisfied by Theorem 3, or (ii) either $r \notin \mathcal{R}(\hat{\mathbf{x}})$ or $\{a, b\} \not\subset V_r$, in which case the right hand side of the constraint (i.e., $Q\hat{x}_{ra}\hat{x}_{rb}$) is zero and the constraint is always satisfied. To show that $\hat{\mathbf{x}}$ satisfies constraint (5.6c) for every $r \in V$, consider two cases: (i) if $r \in \mathcal{R}(\hat{\mathbf{x}})$, the constraint is satisfied since $\hat{x}_{rr} = 1$ and $\sum_{i\in V} \hat{x}_{ri} = |V_r| \geq Q+1$, or (ii) $r \notin \mathcal{R}(\hat{\mathbf{x}})$, and so the right hand side of the constraint (i.e., $(Q+1)\hat{x}_{rr}$) is zero and the constraint is always satisfied.

($\Leftarrow$) We are given that $\hat{\mathbf{x}}$ satisfies constraints (5.6b)-(5.6c). To show that $\hat{\mathbf{x}}$ is Q-proper (or equivalently $\mathcal{P}$ is Q-proper), from Theorem 3, it is sufficient to show that for every $r \in \mathcal{R}(\hat{\mathbf{x}})$, (i) $|V_r| \geq Q+1$, and (ii) for every distinct $a, b \in V_r$ and $C \in \mathcal{C}_{a,b,G}$, $|C \cap V_r| \geq Q$. Constraints (5.6b) imply that for every $r \in \mathcal{R}(\hat{\mathbf{x}})$, for every distinct $a, b \in V_r$ and $C \in \mathcal{C}_{a,b,G}$, $|C \cap V_r| = \sum_{c\in C} \hat{x}_{rc} \geq Q\hat{x}_{ra}\hat{x}_{rb} = Q$. Constraints (5.6c) imply that for every $r \in \mathcal{R}(\hat{\mathbf{x}})$, $|V_r| = \sum_{i\in V} \hat{x}_{ri} \geq (Q+1)\hat{x}_{rr} = Q+1$. □

D.1.3 Proof for Theorem 4

Theorem 4. *Given a graph $G = (V, E)$, an integer $Q \geq 1$, and a solution $\hat{\boldsymbol{x}} \in \mathcal{P}'_{HESS}$, Algorithm 5.1 adds violated separator constraints if and only if $\hat{\boldsymbol{x}}$ is not Q-proper.*

Proof. First, consider the case when $\hat{\mathbf{x}}$ is Q-proper. Equivalently, for every root $r \in R(\hat{\mathbf{x}})$, the subgraph $G[V_r]$ is Q-connected $\Rightarrow$ the size of every cutset of $G[V_r]$ is at least Q and $|V_r| \geq Q+1$. Since $Q \geq 1$ implies that $G[V_r]$ is connected, the single component of $G[V_r]$ must contain r, and so lines 2-5 are skipped. In line 6, this single component is termed G_r, which is exactly $G[V_r]$. There are two cases for the set D returned in line 7: either (i) $G[V_r]$ is a complete graph, in which case the algorithm returns $D = V_r\backslash\{r\}$, and hence $|D| = |V_r| - 1 \geq (Q+1) - 1 = Q$, or (ii) $G[V_r]$ is not a complete graph, and the algorithm returns D as a minimum cutset for $G[V_r]$, and hence $|D| \geq Q$ since $G[V_r]$ is Q-connected. In either case, we have $|D| \geq Q$, and lines 9-18 are skipped. Therefore, Algorithm 5.1 does not add any violated Q-connectivity constraints.

Next, consider the case when $\hat{\mathbf{x}}$ is not Q-proper. For the rest of this proof, consider an arbitrary root $r \in R(\hat{\mathbf{x}})$ such that $G[V_r]$ is not Q-connected. Note that $G[V_r]$ is not a complete graph since $|V_r| \geq Q+1$ from constraints (5.8); a complete graph would have connectivity $|V_r| - 1 \geq Q$ and hence be Q-connected. First, we show that $\hat{\mathbf{x}}$ violates at least one of the three sets of constraints added in lines 5, 13, and 18. If $G[V_r]$ is disconnected, we show that $\hat{\mathbf{x}}$ violates the constraints added in line 5; else, it violates either the

constraints added in line 13 or the ones in line 18. Then, we show that these sets of constraints are subsets of constraints (5.6b) and (5.7a)-(5.7c).

The first set of constraints is added (in line 5) if $G[V_r]$ is disconnected, i.e., there exists a component G_0 of $G[V_r]$ that does not contain r. In this case, line 3 first selects a vertex b in G_0. Line 4 finds a minimal r, b-separator C. We now show that none of the vertices in C are assigned to root r, i.e., $C \cap V_r = \emptyset$. Consider the contrary, that there exists some vertex $u \in C \cap V_r$. Line 23 implies that $u \in N(V[G_0])$, and hence does not belong to G_0, but is a neighbor of the vertices in G_0. The fact that $u \in V_r$ is a contradiction since G_0 is a component of $G[V_r]$. Hence, $C \cap V_r = \emptyset$, and therefore $\hat{x}_{rc} = 0$ for all $c \in C$. In the constraint in line 5 (i.e., $Q\hat{x}_{rb} \leq \sum_{c \in C} \hat{x}_{rc}$), since $\hat{x}_{rb} = 1$, and the right-hand side is zero, this constraint is violated.

The second set of constraints is added (in line 13) if $|D| < Q$ and $r \notin D$. Here, line 11 first selects a vertex b in a component G_0 of $G_r - D$. Subsequently, line 12 finds a minimal r, b-separator C in G such that $C \subseteq N(V[G_0])$. To compute $\sum_{c \in C} \hat{x}_{rc}$, we must identify the vertices of C that are assigned to root r (i.e., those with $\hat{x}_{rc} = 1$). For any $u \in C$, there are two cases:

- $u \in D$: Since $D \subseteq V_r$, we have $u \in V_r$.
- $u \notin D$: Recall that $C \subseteq N(V[G_0])$, and hence u has at a neighbor $v \in V[G_0] \subseteq V_r$. Since G_0 is a component of $G[V_r]$, then if both u and v are in V_r, they must be in the same component of $G[V_r]$; this cannot occur, since we cannot have both $u \in V[G_0]$ and $u \in N(V[G_0])$. Hence, we have $u \notin V_r$.

Hence, $C \cap V_r \subseteq D$, and therefore $\sum_{c \in C} \hat{x}_{rc} \leq |D| < Q$. In the constraint in line 13 (i.e., $Q\hat{x}_{rb} \leq \sum_{c \in C} \hat{x}_{rc}$), since $\hat{x}_{rb} = 1$, and the right-hand side is less than Q, this constraint is violated.

The third set of constraints is added (in line 18) if $|D| < Q$ and $r \in D$. Here, line 16 first selects vertices a and b from components G_1 and G_2, respectively, of $G_r - D$. Line 17 finds a minimal a, b-separator C in G. First, we see that $C \cap V_r \subseteq D$ using the same proof by contradiction for the second set of constraints. Therefore, $\sum_{c \in C} \hat{x}_{rc} \leq |D| < Q$. In the constraint in line 18 (i.e., $Q(\hat{x}_{ra} + \hat{x}_{rb} - 1) \leq \sum_{c \in C} \hat{x}_{rc}$), since $\hat{x}_{ra} = \hat{x}_{rb} = 1$, and the right-hand side is less than Q, this constraint is violated.

We now show that each of the three sets of constraints is a subset of constraints (5.6b). For the first and second sets of constraints added in lines 5 and 13, respectively, the claim holds when we set $a = r$ and note that $\hat{x}_{ra}\hat{x}_{rb} = \hat{x}_{rr}\hat{x}_{rb} = \hat{x}_{rb}$, since r is a root. For the third set of constraints added in line 18, the claim holds from substituting $\hat{x}_{ra}\hat{x}_{rb}$ with y^r_{ab} in constraint (5.6b), and then combining with constraint (5.7c). Hence, all three sets of constraints are Q-connectivity violated constraints. □

D.1.4 Proof for Theorem 5

Theorem 5. *Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, integers $Q \geq 1$ and $K \geq 2$, and a solution $\hat{\boldsymbol{x}} \in \mathcal{P}'_{HESS}$, Algorithm 5.1 takes $O(Kmn^{5/3})$ time if all the parts of $\hat{\boldsymbol{x}}$ are root-resilient to the cutsets in line 8 or $K \geq \sqrt[3]{n}$; otherwise, it takes $O(mn^2)$ time.*

Proof. First, finding a minimal separator from Fischetti et al., 2017 takes $O(m)$ time (Validi et al., 2022). Since every vertex b (in line 3) can be visited at most once among all roots r and all components G_0, lines 1-5 take $O(mn)$ time. Next, in lines 6 and 7, finding a minimum cutset in G_r takes $O(mn^{5/3})$ time using Algorithm 11 in Esfahanian, 2013. Lines 6 and 7 take $O(Kmn^{5/3})$ time when iterating over all the K roots. Next, in lines 9-13, every vertex b can be visited at most once since each b is present in exactly one component of $G_r - D$ among all roots r. Hence, lines 9-13 take $O(mn)$ time using the same reasoning for lines 1-5. If all the parts are root-resilient, only lines 1-13 are executed, and hence the overall run-time is $O(Kmn^{5/3})$.

If some part is not root-resilient, lines 15-18 are also executed. Here, each pair of vertices a and b are visited exactly once among all roots r and all pairs of components G_1 and G_2, resulting in $O(n^2)$ iterations of the for-loop in line 15, where each iteration takes $O(m)$ to generate a minimal separator and add the cut. Lines 15-18 take $O(mn^2)$ time. Hence, if some parts are not root-resilient, executing lines 1-18 takes $O(Kmn^{5/3} + mn^2)$ time, which is $O(Kmn^{5/3})$ time if $K \geq \sqrt[3]{n}$; otherwise, it takes $O(mn^2)$ time. □

D.2 Pseudocodes for the Ear Construction Heuristic

This appendix provides pseudocodes for the ear construction heuristic from Section 5.5.

D.2.1 Overview

Algorithm D.1: Ear-Construction Heuristic (G, K, L, U)

```
1  P ← Construct an Initial Partition (G, K, L)
2  if P is not 2-proper then
3  |  P ← Repair 2-Connectivity (G, P)
4  |  if P is not 2-proper then
5  |  |  Go to line 1
6  if P is not L,U-balanced then
7  |  P ← Local Search (G, P, φ_bal)
8  |  if P is not L,U-balanced then
9  |  |  Go to line 1
10 P ← Local Search (G, P, φ_comp)
11 return P
```

Algorithm D.1 provides an overview of the heuristic. Line 1 executes the construct stage described in Section 5.5.2, lines 2-5 execute the repair stage described in Section 5.5.3, and lines 6-10 execute the improvement stages described in Section 5.5.4. The pseudocodes for these stages are provided in the following subsections.

D.2.2 Ear Construction

Algorithm D.2: Ear Construction (H, L)

```
1  P_0 ← Find a cycle in H
2  T ← P_0: Set of traversed vertices
3  while size(T) < L do
4  |  T^C ← V[H]\T: Set of untraversed vertices
5  |  Randomly shuffle T
6  |  for (u, v) ∈ T × T with u ≠ v do
7  |  |  Assign H_{u,v} ← H[T^C ∪ {u, v}] and remove edge (u, v) from H_{u,v}
8  |  |  if there exists a u, v-path in H_{u,v} then
9  |  |  |  P ← Shortest u, v-path in H_{u,v}
10 |  |  |  T ← T ∪ P
11 |  |  |  Go to line 3
12 |  if no u, v-path exists in H_{u,v} for any (u, v) ∈ T × T with u ≠ v then
13 |  |  return T
14 return T
```

Algorithm D.2 outlines a pseudocode to construct a 2-connected subgraph of a graph H as described in Section 5.5.1. This procedure follows the construction procedure in Definition 3. Line 1 finds a random cycle as follows. First, the procedure randomly selects two neighboring vertices u and v in H and finds a shortest path in H after removing the edge (u, v); if no such path exists, this process is repeated with different pairs of neighboring vertices. The vertices traversed in the resulting path are assigned as the cycle P_0. If H does not contain any cycles (i.e., $P_0 = \emptyset$), indicating that it is a forest, the algorithm returns an empty graph since no 2-connected subgraph exists. If a cycle is identified, lines 3-11 iteratively search for open ears and add them to the traversal. In each iteration, consistent with the procedure in Definition 3, the procedure first selects a pair of vertices u and v in the traversed vertices such that there is a u, v-path in the remaining graph. If such a path exists, the procedure finds the shortest vertex-weighted u, v-path and add those vertices to the traversal; this path must include at least one vertex from $V[H]\backslash T$, since the edge (u, v) does not appear in $H_{u,v}$. This procedure continues until either the total size of the traversed vertices is at least L, or if no more open ears can be added. Hence, Algorithm D.2 either returns the set of vertices in a 2-connected subgraph of H or an empty set.

D.2.3 Construct an Initial Partition

Algorithm D.3 outlines a pseudocode for the procedure described in Section 5.5.2. Throughout this procedure, let G' (initialized with G) denote the remaining graph. Lines 3-6 iteratively find a set of 2-connected subgraphs using Algorithm D.2. After these iterations, if G' is a forest, lines 7-9 designate each tree in the forest as an individual part. If $K' < K$, line 11 directs the algorithm to restart. If $K' > K$, line 13 proceeds to merge parts iteratively until $K' = K$. Finally, Algorithm D.3 returns a partition with K connected parts, some of which may be 2-disconnected.

Algorithm D.3: Construct an Initial Partition (G, K, L)

1 $G'(V', E') \leftarrow G(V, E)$: Remaining graph
2 $K' \leftarrow 0$: Initiate the label for the current part
3 **while** G' *contains a cycle* **do**
4 | $K' \leftarrow K' + 1$
5 | $V^{(K')} \leftarrow$ Ear Construction(G', L)
6 | $G' \leftarrow G' - V^{(K')}$
7 **for** *tree* T *of* G' **do**
8 | $K' \leftarrow K' + 1$
9 | $V^{(K')} \leftarrow T$
10 **if** $K' < K$ **then**
11 | Go to Line 1
12 **while** $K' > K$ **do**
13 | Select two neighboring parts and merge; proceed in the non-increasing order of pair-wise total part sizes
14 **return** $\{V^{(k)}\}_{k=1}^{K}$

D.2.4 Repair 2-Connectivity

Algorithm D.4: Repair 2-Connectivity (G, $\{V^{(k)}\}_{k=1}^{K}$)

1 **while** $\{V^{(k)}\}_{k=1}^{K}$ *is not* 2*-proper* **do**
2 | $k \leftarrow$ a random 2-disconnected part
3 | $c \leftarrow$ a random cut vertex in $G[V^{(k)}]$
4 | $S \leftarrow$ smallest component of $G[V^{(k)}] - \{c\}$
5 | **if** S *has a neighbor in another part* **then**
6 | | $k' \leftarrow$ a random part neighboring S
7 | | $V^{(k)} \leftarrow V^{(k)} \backslash S$
8 | | $V^{(k')} \leftarrow V^{(k')} \cup S$
9 | **if** S *has no part neighbors or cycling is detected* **then**
10 | | Exit the while loop `// the algorithm restarts from the construct stage`
11 **return** $\{V^{(k)}\}_{k=1}^{K}$

Algorithm D.4 outlines a pseudocode for the repair stage as described in Section 5.5.3. If the given partition is not 2-proper, line 2 randomly selects a 2-disconnected part k, line 3 randomly selects a cut vertex c, and line 4 selects a smallest component S produced by removing the cut vertex; if there are ties, it selects a random such component. The vertices in S are reassigned to a randomly chosen neighboring part k' in lines 6-8. Note that an alternative choice for selecting the neighboring part could be to select a part that maintains 2-connectivity after receiving the vertices, albeit at the expense of needing to recompute each potential recieving part's connectivity. Then, the algorithm assesses a premature termination criterion in line 9, and it exits the loop. Hence, the algorithm either terminates with a 2-proper partition and proceeds to the next stage, or without a 2-proper partition, in which case, the heuristic restarts from the construct stage.

D.2.5 Local Search Improvement

Algorithm D.5: Local Search (G, $\{V^{(k)}\}_{k=1}^{K}$, ϕ)

1 $\phi_{best} \leftarrow \phi(\{V^{(k)}\}_{k=1}^{K})$
2 **while** *termination criterion is not True* **do**
3 $\quad \mathcal{T} \leftarrow$ Randomly ordered set of reassignments
$\quad$ `// each element in` $\mathcal{T}$ `is a vertex` u `and a part` k' `neighboring` u
4 $\quad$ **for** $(u, k') \in \mathcal{T}$ **do**
5 $\quad\quad$ **if** $G[V^{(k)}] - \{u\}$ *and* $G[V^{(k')} \cup \{u\}]$ *are both* 2*-connected* **then**
6 $\quad\quad\quad$ **if** ϕ *improves after moving* u *to* k' **then**
7 $\quad\quad\quad\quad V^{(k)} \leftarrow V^{(k)} \backslash \{u\}$
8 $\quad\quad\quad\quad V^{(k')} \leftarrow V^{(k')} \cup \{u\}$
9 $\quad\quad\quad\quad \phi_{best} \leftarrow \phi(\{V^{(k)}\}_{k=1}^{K})$
10 $\quad\quad\quad\quad$ Go to line 2
11 **return** $\{V^{(k)}\}_{k=1}^{K}$

Algorithm D.5 outlines the local search method described in Section 5.5.4 to optimize an objective $\phi \in \{\phi_{bal}, \phi_{comp}\}$. In each iteration, line 3 creates a random ordering of potential reassignments. Line 4 iterates through each potential reassignment of a vertex u to a neighboring part k'. Line 5 checks whether this reassignment retains 2-connectivity in both parts, and line 6 checks whether this reassignment improves ϕ. Additionally, when optimizing compactness, each iteration must ensure that the size balance objective remains zero. If these conditions are true, lines 7-9 reassign u to k' and update the current best objective value. Furthermore, the termination criterion in line 2 is now described. When optimizing size balance, the algorithm terminates when ϕ_{bal} is zero (i.e., the partition in L, U-balanced), or when no more feasible reassignments improve ϕ_{bal} (i.e., the local search has not balanced the parts, in which case the heuristic will restart from the construct stage). When optimizing compactness, the algorithm terminates when no feasible reassignments improve ϕ_{comp}, i.e., the partition is locally optimal.

D.3 Tables

This appendix provides tabular information on the computational analysis from Section 5.6.

D.3.1 Overview of Instances

Table D.1 depicts an overview of the 42 graph instances, along with their connectivities (κ), average and minimum degrees, as well as the number of vertices removed or edges added in the pre-processing stage.

Table D.1: An overview of 42 graph instances from seven domains with their sizes, densities and the vertices removed/edges added in the pre-processing stage.

Domain	Graph (G)	$\|V\|$	$\|E\|$	$\kappa(G)$	Avg. deg.	Min. deg.	Vertices removed	Edges added
Transportation	Sioux Falls	20	34	2	3.40	3	4	0
Transportation	Berlin-Friedrichshain	158	292	2	3.70	2	66	0
Transportation	Berlin-Prenzlauerberg-Center	214	389	2	3.64	2	138	0
Transportation	Berlin-Tiergarten	280	498	2	3.56	2	79	0
Transportation	Berlin-Mitte-Center	246	451	2	3.67	2	151	0
Transportation	Anaheim	286	486	2	3.40	2	130	0
Transportation	Barcelona	847	1,698	2	4.01	2	83	0
Transportation	Chicago-Sketch	511	1,051	2	4.11	3	422	0
Transportation	Terrassa	874	1,581	2	3.62	2	729	0
Census Tract Adjacency	Rhode Island	222	598	2	5.39	3	22	0
Census Tract Adjacency	New Hampshire	341	921	2	5.40	2	9	0
Census Tract Adjacency	Maine	392	1,100	2	5.61	2	10	0
Census Tract Adjacency	Idaho	437	1,163	2	5.32	2	19	0
Census Tract Adjacency	West Virginia	431	1,180	2	5.48	2	114	0
Census Tract Adjacency	Nebraska	513	1,357	2	5.29	2	40	0
Census Tract Adjacency	New Mexico	576	1,568	2	5.44	2	31	0
Census Tract Adjacency	Utah	706	1,932	2	5.47	2	10	0
Census Tract Adjacency	Arkansas	793	2,186	2	5.51	2	30	0
Census Tract Adjacency	Nevada	764	1,999	2	5.23	2	14	0
Census Tract Adjacency	Mississippi	833	2,272	2	5.45	2	40	0
Census Tract Adjacency	Iowa	833	2,239	2	5.38	2	62	0
Power Networks	IEEE300Bus	247	403	2	3.26	2	0	99
Power Networks	power-494-bus	494	758	2	3.07	2	0	172
Social Networks	3980	52	201	4	7.73	4	0	55
Social Networks	698	61	303	4	9.93	4	0	33
Social Networks	414	150	1,720	4	22.93	4	0	27
Social Networks	686	168	1,696	4	20.19	4	0	40
Social Networks	348	224	3,236	4	28.89	4	0	44
Social Networks	3437	534	4,934	4	18.48	4	0	121
Interaction Networks	hospital-ward-proximity	75	1,139	6	30.37	6	0	0
Interaction Networks	ia-workplace-contacts	92	755	4	16.41	4	0	0
Interaction Networks	infect-hyper	113	2,199	4	38.92	4	0	3
Interaction Networks	ia-radoslaw-email	167	3,334	4	39.93	4	0	83
Interaction Networks	infect-dublin	410	2,843	4	13.87	4	0	78
Animal Social Networks	insecta-ant-colony6-day01	164	10,731	41	130.87	41	0	0
Animal Social Networks	aves-wildbird-network	202	4,583	4	45.38	4	0	9
Animal Social Networks	mammalia-dolphin-florida-overall	291	3,258	4	22.39	4	0	76
Miscellaneous	mycielskian7	95	755	6	15.89	6	0	0
Miscellaneous	mycielskian8	191	2,360	7	24.71	7	0	0
Miscellaneous	mycielskian9	383	7,271	8	37.97	8	0	0
Miscellaneous	qc324	324	13,203	81	81.50	81	0	0
Miscellaneous	ex2	441	13,199	19	59.86	27	0	0

D.3.2 Results for HCGP

Table D.2 presents the distribution of instances for HCGP with $Q = 2$ (among the 252 instances solved in Section 5.6.3), cross-tabulated for the two methods. For example, 23 instances were inconclusive for the exact method and yet feasible (but suboptimal) for the heuristic.

Table D.2: Cross-tabulated distribution of 252 instances for HCGP with $Q = 2$ based on solution outcomes for the two methods: optimal, feasible (and suboptimal), infeasible, or inconclusive.

		Exact method			
		Optimal	Feasible	Infeasible	Inconclusive
Heuristic method	Optimal	9	0	0	0
	Feasible	187	9	0	23
	Infeasible	0	0	0	0
	Inconclusive	8	4	9	3

Tables D.3, D.4 and D.5 present results for HCGP with $Q = 2$, $Q \in \{3, 4\}$, and $Q = 1$, respectively. Each table includes the optimal compactness value (OPT) and the corresponding computational time taken by the branch-and-cut method (OPT time) in seconds. An asterisk (*) or a double asterisk (**) in the OPT column indicates that the solver reached the time limit, thus representing the best known lower bound on the optimal solution. Furthermore, an asterisk denotes that a feasible solution was found by the solver, while a double asterisk indicates that no feasible solution was obtained within the given time limit. If the solver produced an infeasibility certificate for a particular instance, the OPT column displays "inf". The computational times reported in these tables are rounded to the nearest first decimal place; an entry with "0.0" indicates a computational time less than 0.05 seconds.

In Table D.3, the compactness objective of the solution obtained from the ear construction heuristic is reported in the $HEUR$ column. The $HEUR$ time column reports the time taken by the heuristic in seconds. If the heuristic terminated without a feasible solution within the time limit, the $HEUR$ column displays "N.S.", abbreviating "no solution". The approximation gap and the heuristic time ratio are defined in Section 5.6.3. The Approx. Gap column displays "N.A." when either the exact method, the heuristic method, or both, terminated without a feasible solution. When the heuristic found an optimal solution, the Approx. Gap column displays "-".

Table D.3: Results for HCGP with $Q = 2$.

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
Sioux Falls	∞	2	2.79	0.1	2.83	0.0	0.01	12.4
Sioux Falls	∞	3	1.62	0.1	1.67	0.0	0.03	5.4
Sioux Falls	∞	4	1.33	0.0	1.46	0.1	0.10	0.3
Sioux Falls	0.1	2	2.79	0.3	2.83	0.0	0.01	15.9
Sioux Falls	0.1	3	inf	1.9	N.S.	3,600	N.A.	0.0
Sioux Falls	0.1	4	inf	0.7	N.S.	3,600	N.A.	0.0
Berlin-Friedrichshain	∞	2	21.22	7.2	22.83	0.5	0.08	13.4
Berlin-Friedrichshain	∞	3	15.09	8.1	18.00	6.1	0.19	1.3
Berlin-Friedrichshain	∞	4	10.42	12.9	11.99	0.6	0.15	21.2

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
Berlin-Friedrichshain	0.1	2	21.22	35.7	24.71	2.0	0.16	18.1
Berlin-Friedrichshain	0.1	3	15.39	25.9	17.33	317.3	0.13	0.1
Berlin-Friedrichshain	0.1	4	10.71	358.8	11.68	1,590.4	0.09	0.2
Berlin-Prenzlauerberg-Center	∞	2	28.51	11.0	34.48	1.0	0.21	11.6
Berlin-Prenzlauerberg-Center	∞	3	18.01	10.2	20.88	0.6	0.16	15.7
Berlin-Prenzlauerberg-Center	∞	4	12.74	29.0	20.41	1.1	0.60	26.0
Berlin-Prenzlauerberg-Center	0.1	2	28.51	123.9	35.09	104.3	0.23	1.2
Berlin-Prenzlauerberg-Center	0.1	3	18.33	66.1	20.22	71.6	0.10	0.9
Berlin-Prenzlauerberg-Center	0.1	4	12.74	85.4	16.70	817.6	0.31	0.1
Berlin-Tiergarten	∞	2	34.31	32.7	37.28	1.6	0.09	20.3
Berlin-Tiergarten	∞	3	22.36	43.3	29.09	1.2	0.30	35.9
Berlin-Tiergarten	∞	4	17.60	90.3	30.14	8.7	0.71	10.4
Berlin-Tiergarten	0.1	2	34.31	267.3	N.S.	3,600	N.A.	0.1
Berlin-Tiergarten	0.1	3	24.58*	3,600	N.S.	3,600	N.A.	1
Berlin-Tiergarten	0.1	4	18.96**	3,600	N.S.	3,600	N.A.	1
Berlin-Mitte-Center	∞	2	25.07	17.7	28.59	1.0	0.14	17.1
Berlin-Mitte-Center	∞	3	17.82	20.1	22.86	0.9	0.28	22.0
Berlin-Mitte-Center	∞	4	12.46	16.5	19.92	2.6	0.60	6.3
Berlin-Mitte-Center	0.1	2	25.45	93.0	36.89	10.5	0.45	8.9
Berlin-Mitte-Center	0.1	3	19.01	372.9	19.51	576.2	0.03	0.6
Berlin-Mitte-Center	0.1	4	13.39	465.6	14.76	1,973.5	0.10	0.2
Anaheim	∞	2	27.48	16.5	34.22	1.3	0.25	13.0
Anaheim	∞	3	18.69	31.8	24.56	1.0	0.31	31.9
Anaheim	∞	4	13.37	26.1	18.42	1.5	0.38	17.0
Anaheim	0.1	2	27.48	277.1	31.56	9.8	0.15	28.3
Anaheim	0.1	3	19.07	224.3	21.87	160.6	0.15	1.4
Anaheim	0.1	4	13.69	225.7	N.S.	3,600	N.A.	0.1
Barcelona	∞	2	57.11	711.6	68.73	11.8	0.20	60.2
Barcelona	∞	3	36.87	529.5	56.40	9.7	0.53	54.6
Barcelona	∞	4	28.76	671.0	42.77	22.4	0.49	30.0
Barcelona	0.1	2	0.0**	3,600	92.52	87.5	N.A.	41.1
Barcelona	0.1	3	36.87	862.5	52.76	117.7	0.43	7.3
Barcelona	0.1	4	0.0**	3,600	39.60	2,711.7	N.A.	1.3
Chicago-Sketch	∞	2	43.34	71.8	47.28	4.4	0.09	16.5
Chicago-Sketch	∞	3	28.66	53.2	42.83	6.4	0.49	8.3
Chicago-Sketch	∞	4	22.18	71.5	36.96	2.6	0.67	27.8
Chicago-Sketch	0.1	2	43.34	342.8	64.31	7.2	0.48	47.7
Chicago-Sketch	0.1	3	29.05	2,095.4	41.38	75.2	0.42	27.9
Chicago-Sketch	0.1	4	22.20	116.7	32.04	420.3	0.44	0.3
Terrassa	∞	2	84.07	302.9	128.22	39.5	0.53	7.7
Terrassa	∞	3	53.59	385.6	73.63	7.6	0.37	50.7
Terrassa	∞	4	39.5*	3,600	50.66	11.3	0.28	318.1
Terrassa	0.1	2	84.07	508.0	112.25	51.9	0.34	9.8

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
Terrassa	0.1	3	0.0**	3,600	57.79	31.6	N.A.	113.8
Terrassa	0.1	4	0.0**	3,600	48.27	3,114.4	N.A.	1.2
Rhode Island	∞	2	13.05	14.2	13.42	1.6	0.03	9.1
Rhode Island	∞	3	8.40	9.4	9.64	1.5	0.15	6.1
Rhode Island	∞	4	6.29	12.4	14.87	3.3	1.36	3.8
Rhode Island	0.1	2	13.08	54.1	14.15	4.8	0.08	11.3
Rhode Island	0.1	3	8.64	84.6	9.99	28.2	0.16	3.0
Rhode Island	0.1	4	6.31	112.2	8.65	47.4	0.37	2.4
New Hampshire	∞	2	20.18	30.4	23.63	9.9	0.17	3.1
New Hampshire	∞	3	12.93	28.2	13.33	6.0	0.03	4.7
New Hampshire	∞	4	9.97	22.3	12.30	5.6	0.23	4.0
New Hampshire	0.1	2	20.42	200.4	23.46	10.3	0.15	19.4
New Hampshire	0.1	3	13.04	1,074.9	15.33	9.1	0.18	118.4
New Hampshire	0.1	4	9.97	917.4	11.68	21.8	0.17	42.1
Maine	∞	2	17.36	37.9	21.98	8.0	0.27	4.7
Maine	∞	3	13.30	120.4	15.82	5.9	0.19	20.3
Maine	∞	4	10.19	80.2	12.26	16.8	0.20	4.8
Maine	0.1	2	17.64	795.6	18.98	10.9	0.08	73.1
Maine	0.1	3	13.82**	3,600	14.72	36.4	N.A.	99.0
Maine	0.1	4	10.24	3,255	14.23	69.2	0.39	47.1
Idaho	∞	2	19.95	53.7	22.43	12.6	0.12	4.3
Idaho	∞	3	12.94	89.3	19.55	3.2	0.51	27.9
Idaho	∞	4	9.84	42.0	14.59	4.2	0.48	10.0
Idaho	0.1	2	20.96	443.0	N.S.	3,600	N.A.	0.1
Idaho	0.1	3	14.55**	3,600	16.52	48.9	N.A.	73.6
Idaho	0.1	4	10.26**	3,600	N.S.	3,600	N.A.	1
West Virginia	∞	2	25.31	53.1	28.59	6.3	0.13	8.4
West Virginia	∞	3	16.94	59.8	23.11	6.4	0.36	9.3
West Virginia	∞	4	11.84	44.3	12.88	10.2	0.09	4.3
West Virginia	0.1	2	25.37	784.1	26.06	11.1	0.03	70.6
West Virginia	0.1	3	17.00	196.4	20.57	18.6	0.21	10.6
West Virginia	0.1	4	12.08	280.6	16.02	26.2	0.33	10.7
Nebraska	∞	2	22.06	66.4	28.86	9.6	0.31	6.9
Nebraska	∞	3	14.78	73.1	19.12	4.1	0.29	17.6
Nebraska	∞	4	11.57	53.3	17.46	5.8	0.51	9.1
Nebraska	0.1	2	26.33**	3,600	28.83	15.7	N.A.	229.7
Nebraska	0.1	3	15.67	854.3	19.63	121.2	0.25	7.0
Nebraska	0.1	4	11.90	3,239.6	15.91	88.5	0.34	36.6
New Mexico	∞	2	17.25	80.9	23.99	22.3	0.39	3.6
New Mexico	∞	3	12.92	101.6	15.68	14.2	0.21	7.2
New Mexico	∞	4	11.16	126.0	14.07	8.5	0.26	14.8
New Mexico	0.1	2	0.0**	3,600	19.69	39.8	N.A.	90.5
New Mexico	0.1	3	13.05	627.4	15.99	343.2	0.23	1.8

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
New Mexico	0.1	4	11.65**	3,600	N.S.	3,600	N.A.	1
Utah	∞	2	17.50	161.2	22.73	20.7	0.30	7.8
Utah	∞	3	14.52	2,778.0	19.70	24.0	0.36	115.7
Utah	∞	4	11.73	1,758.3	15.51	18.5	0.32	95.0
Utah	0.1	2	17.50	1,255.3	23.14	25.5	0.32	49.3
Utah	0.1	3	15.36**	3,600	20.67	553.2	N.A.	6.5
Utah	0.1	4	12.85**	3,600	16.34	31.0	N.A.	116.1
Arkansas	∞	2	35.28	263.2	37.41	34.5	0.06	7.6
Arkansas	∞	3	23.86	274.7	30.07	19.1	0.26	14.3
Arkansas	∞	4	17.83	227.5	25.93	29.7	0.45	7.6
Arkansas	0.1	2	35.28**	3,600	47.48	117.9	N.A.	30.5
Arkansas	0.1	3	24.03**	3,600	25.49	232.6	N.A.	15.5
Arkansas	0.1	4	0.0**	3,600	20.77	112.1	N.A.	32.1
Nevada	∞	2	23.23	1,000.6	31.77	12.7	0.37	79.0
Nevada	∞	3	16.15	255.4	30.32	17.0	0.88	15.0
Nevada	∞	4	13.45	393.4	27.27	14.9	1.03	26.5
Nevada	0.1	2	0.0**	3,600	32.98	34.4	N.A.	104.8
Nevada	0.1	3	17.79**	3,600	25.83	112.4	N.A.	32.0
Nevada	0.1	4	13.69**	3,600	19.41	134.5	N.A.	26.8
Mississippi	∞	2	39.38	225.3	43.92	16.9	0.12	13.4
Mississippi	∞	3	25.23	251.1	39.94	15.7	0.58	16.0
Mississippi	∞	4	20.85	750.2	24.69	19.1	0.18	39.3
Mississippi	0.1	2	0.0**	3,600	41.04	234.7	N.A.	15.3
Mississippi	0.1	3	0.0**	3,600	26.28	57.9	N.A.	62.1
Mississippi	0.1	4	0.0**	3,600	23.92	103.3	N.A.	34.8
Iowa	∞	2	33.21	258.6	37.52	35.1	0.13	7.4
Iowa	∞	3	24.38	219.3	36.51	51.9	0.50	4.2
Iowa	∞	4	19.56	648.9	27.43	31.1	0.40	20.8
Iowa	0.1	2	0.0**	3,600	40.25	344.6	N.A.	10.4
Iowa	0.1	3	0.0**	3,600	28.79	150.6	N.A.	23.9
Iowa	0.1	4	0.0**	3,600	23.10	2,319.6	N.A.	1.6
IEEE300Bus	∞	2	6.04	110.1	6.55	1,118.7	0.08	0.1
IEEE300Bus	∞	3	5.86	94.5	N.S.	3,600	N.A.	0.0
IEEE300Bus	∞	4	5.83	115.0	N.S.	3,600	N.A.	0.0
IEEE300Bus	0.1	2	inf	6.1	N.S.	3,600	N.A.	0.0
IEEE300Bus	0.1	3	inf	5.0	N.S.	3,600	N.A.	0.0
IEEE300Bus	0.1	4	inf	4.5	N.S.	3,600	N.A.	0.0
power-494-bus	∞	2	6.55	598.3	N.S.	3,600	N.A.	0.2
power-494-bus	∞	3	6.43	635.2	N.S.	3,600	N.A.	0.2
power-494-bus	∞	4	inf	2,560.6	N.S.	3,600	N.A.	0.7
power-494-bus	0.1	2	inf	86.6	N.S.	3,600	N.A.	0.0
power-494-bus	0.1	3	inf	82.8	N.S.	3,600	N.A.	0.0
power-494-bus	0.1	4	inf	79.3	N.S.	3,600	N.A.	0.0

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
3980	∞	2	1.48	0.2	1.88	0.2	0.27	0.9
3980	∞	3	1.17	0.8	1.46	0.2	0.25	5.1
3980	∞	4	1.04	0.5	1.50	0.2	0.44	2.7
3980	0.1	2	1.48	0.2	1.94	4.5	0.31	0.0
3980	0.1	3	1.35	1.9	2.23	17.8	0.65	0.1
3980	0.1	4	1.15	1.4	N.S.	3,600	N.A.	0.0
698	∞	2	1.80	0.9	1.85	0.4	0.03	2.4
698	∞	3	1.49	1.1	1.84	0.5	0.23	2.0
698	∞	4	1.23	0.6	1.57	0.4	0.28	1.5
698	0.1	2	1.80	0.8	1.90	0.6	0.06	1.3
698	0.1	3	1.64	5.4	2.46	13.8	0.50	0.4
698	0.1	4	1.43	3.9	1.52	123.5	0.06	0.0
414	∞	2	1.89	6.5	2.52	13.5	0.33	0.5
414	∞	3	1.42	7.3	2.19	7.8	0.54	0.9
414	∞	4	1.25	15.4	1.53	7.2	0.22	2.1
414	0.1	2	2.21	18.5	2.35	11.5	0.06	1.6
414	0.1	3	1.68	17.0	2.02	10.2	0.20	1.7
414	0.1	4	1.43	99.4	1.63	26.6	0.14	3.7
686	∞	2	2.17	7.9	2.35	12.2	0.08	0.6
686	∞	3	1.86	20.4	2.45	10.0	0.32	2.0
686	∞	4	1.57	11.7	2.29	8.1	0.46	1.5
686	0.1	2	2.17	14.6	2.46	18.1	0.13	0.8
686	0.1	3	1.86	45.3	2.72	15.2	0.46	3.0
686	0.1	4	1.67	83.4	2.09	32.0	0.25	2.6
348	∞	2	2.25	39.8	2.42	33.9	0.08	1.2
348	∞	3	1.76	14.5	2.37	24.5	0.35	0.6
348	∞	4	1.58	24.1	2.15	36.9	0.36	0.7
348	0.1	2	2.25	117.9	2.42	46.8	0.08	2.5
348	0.1	3	1.76	212.2	2.42	72.2	0.38	2.9
348	0.1	4	1.58	280.9	2.55	72.8	0.61	3.9
3437	∞	2	4.34	229.0	5.60	62.6	0.29	3.7
3437	∞	3	3.40	353.1	5.20	72.8	0.53	4.9
3437	∞	4	2.85**	3,600	4.23	62.1	N.A.	58.0
3437	0.1	2	4.34	272.7	5.22	50.1	0.20	5.4
3437	0.1	3	3.40	346.6	4.89	164.9	0.44	2.1
3437	0.1	4	2.87*	3,600	4.68	638.9	0.63	5.6
hospital-ward-proximity	∞	2	1.13	1.7	1.25	3.4	0.11	0.5
hospital-ward-proximity	∞	3	1.00	1.9	1.28	2.9	0.28	0.7
hospital-ward-proximity	∞	4	0.95	1.2	1.27	2.6	0.34	0.4
hospital-ward-proximity	0.1	2	1.13	1.9	1.25	5.7	0.11	0.3
hospital-ward-proximity	0.1	3	1.00	5.1	1.16	6.1	0.16	0.8
hospital-ward-proximity	0.1	4	0.95	3.3	1.23	5.1	0.29	0.6
ia-workplace-contacts	∞	2	1.73	0.8	2.02	1.8	0.17	0.4

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
ia-workplace-contacts	∞	3	1.29	0.8	2.01	1.2	0.56	0.7
ia-workplace-contacts	∞	4	1.15	1.2	2.04	1.6	0.77	0.7
ia-workplace-contacts	0.1	2	1.73	1.3	2.21	2.7	0.28	0.5
ia-workplace-contacts	0.1	3	1.29	1.1	2.21	2.4	0.71	0.5
ia-workplace-contacts	0.1	4	1.15	1.5	2.32	4.7	1.02	0.3
infect-hyper	∞	2	1.06	2.7	1.27	12.0	0.20	0.2
infect-hyper	∞	3	0.97	6.0	1.32	9.7	0.36	0.6
infect-hyper	∞	4	0.96	5.3	1.20	10.2	0.25	0.5
infect-hyper	0.1	2	1.06	16.3	1.30	23.8	0.23	0.7
infect-hyper	0.1	3	0.97	4.3	1.13	21.1	0.16	0.2
infect-hyper	0.1	4	0.96	7.7	1.34	12.3	0.40	0.6
ia-radoslaw-email	∞	2	0.99	10.0	1.04	26.6	0.05	0.4
ia-radoslaw-email	∞	3	0.98	4.9	1.05	22.8	0.07	0.2
ia-radoslaw-email	∞	4	0.98	6.1	1.12	18.9	0.14	0.3
ia-radoslaw-email	0.1	2	0.99	11.4	1.04	45.2	0.05	0.3
ia-radoslaw-email	0.1	3	0.98	8.8	1.16	32.7	0.18	0.3
ia-radoslaw-email	0.1	4	0.98	19.7	1.21	36.1	0.23	0.5
infect-dublin	∞	2	3.75	82.2	5.70	22.1	0.52	3.7
infect-dublin	∞	3	3.19	49.5	3.69	11.8	0.16	4.2
infect-dublin	∞	4	2.84	60.7	3.73	15.4	0.31	3.9
infect-dublin	0.1	2	3.79	84.1	5.70	45.2	0.50	1.9
infect-dublin	0.1	3	3.19	174.8	5.21	122.2	0.63	1.4
infect-dublin	0.1	4	2.84	111.8	4.09	222.7	0.44	0.5
insecta-ant-colony6-day01	∞	2	0.99	33.6	0.99	179.4	-	0.2
insecta-ant-colony6-day01	∞	3	0.98	21.4	0.98	145.2	-	0.1
insecta-ant-colony6-day01	∞	4	0.98	27.8	0.98	117.4	-	0.2
insecta-ant-colony6-day01	0.1	2	0.99	44.6	0.99	291.7	-	0.2
insecta-ant-colony6-day01	0.1	3	0.98	32.0	0.98	244.1	-	0.1
insecta-ant-colony6-day01	0.1	4	0.98	20.6	0.98	192.0	-	0.1
aves-wildbird-network	∞	2	1.55	19.5	1.78	47.8	0.15	0.4
aves-wildbird-network	∞	3	1.34	251.8	1.46	60.2	0.09	4.2
aves-wildbird-network	∞	4	1.14	24.0	1.56	64.2	0.37	0.4
aves-wildbird-network	0.1	2	1.60	33.2	1.81	102.3	0.13	0.3
aves-wildbird-network	0.1	3	1.34	30.1	1.50	75.2	0.12	0.4
aves-wildbird-network	0.1	4	1.14	63.3	1.53	315.4	0.34	0.2
mammalia-dolphin-florida-overall	∞	2	2.46	47.5	2.74	38.6	0.11	1.2
mammalia-dolphin-florida-overall	∞	3	2.06	45.4	3.03	36.2	0.47	1.3
mammalia-dolphin-florida-overall	∞	4	1.83	56.2	2.66	33.4	0.45	1.7
mammalia-dolphin-florida-overall	0.1	2	2.46	53.1	2.82	53.2	0.15	1.0
mammalia-dolphin-florida-overall	0.1	3	2.06	980.3	2.95	63.6	0.43	15.4
mammalia-dolphin-florida-overall	0.1	4	1.83	80.1	2.62	110.3	0.43	0.7
mycielskian7	∞	2	1.74	8.8	2.12	0.3	0.22	33.9
mycielskian7	∞	3	1.38	53.6	2.23	0.2	0.62	290.8

Table D.3: Results for HCGP with $Q = 2$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)	$HEUR$	$HEUR$ time (s)	Approx. Gap	Heur. time ratio
mycielskian7	∞	4	1.24	263.9	2.19	0.2	0.77	1,688.5
mycielskian7	0.1	2	1.74	18.6	2.12	0.4	0.22	47.3
mycielskian7	0.1	3	1.38	78.1	2.33	18.1	0.69	4.3
mycielskian7	0.1	4	1.22*	3,600	N.S.	3,600	N.A.	1
mycielskian8	∞	2	1.74	232.9	2.28	1.3	0.31	180.6
mycielskian8	∞	3	1.38	680.3	2.46	0.8	0.78	859.1
mycielskian8	∞	4	1.07*	3,600	2.24	0.7	1.09	5,187.1
mycielskian8	0.1	2	1.74	363.0	2.15	5.7	0.24	63.2
mycielskian8	0.1	3	1.38	1,547.9	2.35	130.7	0.70	11.8
mycielskian8	0.1	4	1.14*	3,600	N.S.	3,600	N.A.	1
mycielskian9	∞	2	1.75	611.9	2.24	6.4	0.28	96.0
mycielskian9	∞	3	1.19*	3,600	2.43	3.9	1.04	916.5
mycielskian9	∞	4	1.04*	3,600	2.36	3.5	1.27	1,036.6
mycielskian9	0.1	2	1.75	1,701.8	2.79	293.4	0.59	5.8
mycielskian9	0.1	3	1.18*	3,600	2.46	97.4	1.08	37.0
mycielskian9	0.1	4	1.05*	3,600	N.S.	3,600	N.A.	1
qc324	∞	2	2.47	856.0	2.48	162.0	0.00	5.3
qc324	∞	3	1.7*	3,600	1.97	143.6	0.16	25.1
qc324	∞	4	0.99	37.9	0.99	213.6	-	0.2
qc324	0.1	2	2.46*	3,600	2.48	271.5	0.01	13.3
qc324	0.1	3	1.7*	3,600	1.95	420.2	0.15	8.6
qc324	0.1	4	0.99	745.9	0.99	398.9	-	1.9
ex2	∞	2	2.33	88.4	2.36	173.5	0.01	0.5
ex2	∞	3	1.85	125.6	1.88	292.3	0.02	0.4
ex2	∞	4	1.51	53.2	1.51	287.6	-	0.2
ex2	0.1	2	2.33	499.7	2.36	278.6	0.01	1.8
ex2	0.1	3	1.85	916.1	2.11	366.3	0.14	2.5
ex2	0.1	4	1.51	61.6	1.86	424.0	0.23	0.1

In Table D.4, the column "Callback time" reports the total time spent in the callbacks, measured in seconds. When the branch-and-cut method does not enter any callback, this column reports a "-". Further, the "Conn." column reports the individual connectivities of the K parts.

Table D.4: Results for HCGP with $Q \in \{3, 4\}$.

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
3980	3	∞	2	1.60	1.5	0.8	3,3
3980	3	∞	3	1.46	3.2	0.8	3,3,3
3980	3	∞	4	1.33	2.2	0.3	3,3,3,3
3980	3	0.1	2	2.19	12.6	3.6	3,3

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
3980	3	0.1	3	inf	83.3	-	N.A.
3980	3	0.1	4	inf	77.5	-	N.A.
3980	4	∞	2	inf	0.1	-	N.A.
3980	4	∞	3	inf	0.1	-	N.A.
3980	4	∞	4	inf	0.1	-	N.A.
3980	4	0.1	2	inf	0.1	-	N.A.
3980	4	0.1	3	inf	0.1	-	N.A.
3980	4	0.1	4	inf	0.1	-	N.A.
698	3	∞	2	1.80	0.8	0.5	3,3
698	3	∞	3	1.54	0.7	0.4	3,3,3
698	3	∞	4	1.28	0.4	0.1	3,3,6,3
698	3	0.1	2	1.80	0.7	0.4	3,3
698	3	0.1	3	2.49**	3,600	18.9	N.A.
698	3	0.1	4	2.23**	3,600	13.4	N.A.
698	4	∞	2	2.25	3.2	1.2	4,4
698	4	∞	3	1.84	1.6	0.3	4,5,4
698	4	∞	4	1.82	1.5	0.2	4,4,4,4
698	4	0.1	2	inf	671.9	40.2	N.A.
698	4	0.1	3	4.25**	3,600	27.7	N.A.
698	4	0.1	4	2.89**	3,600	28.9	N.A.
414	3	∞	2	1.91	6.5	2.2	3,4
414	3	∞	3	1.44	10.1	3.4	11,3,4
414	3	∞	4	1.27	17.7	11.3	11,3,3,4
414	3	0.1	2	2.29	21.0	4.1	3,3
414	3	0.1	3	1.74	42.3	5.6	3,3,4
414	3	0.1	4	1.57	1,517.4	79.4	3,3,3,3
414	4	∞	2	2.02	18.8	4.5	4,4
414	4	∞	3	1.55	18.2	3.7	11,4,4
414	4	∞	4	1.45	48.9	4.0	11,4,4,4
414	4	0.1	2	2.53	21.3	2.7	4,4
414	4	0.1	3	2.71**	3,600	536.1	N.A.
414	4	0.1	4	2.12**	3,600	282.3	N.A.
686	3	∞	2	2.22	13.4	4.3	3,3
686	3	∞	3	1.91	25.9	5.5	3,3,3
686	3	∞	4	1.67	32.8	3.2	3,3,3,3
686	3	0.1	2	2.22	18.4	1.9	3,3
686	3	0.1	3	1.95	54.9	3.8	4,3,3
686	3	0.1	4	1.77	205.3	3.9	3,3,3,3
686	4	∞	2	2.43	85.1	10.0	4,4
686	4	∞	3	2.14	157.3	2.7	4,4,4
686	4	∞	4	1.97	212.6	3.5	4,4,4,4
686	4	0.1	2	2.43	49.8	6.5	4,4
686	4	0.1	3	inf	1.9	-	N.A.

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
686	4	0.1	4	inf	1.8	-	N.A.
348	3	∞	2	2.29	35.6	11.8	3,3
348	3	∞	3	1.80	26.7	8.2	3,3,3
348	3	∞	4	1.64	40.5	9.2	3,3,3,3
348	3	0.1	2	2.29	131.2	11.8	3,3
348	3	0.1	3	1.80	365.4	3.7	3,3,3
348	3	0.1	4	1.64	129.6	6.2	3,3,3,3
348	4	∞	2	2.37	87.6	30.6	4,4
348	4	∞	3	2.01	189.6	29.7	4,4,4
348	4	∞	4	1.81	272.8	9.6	4,4,4,4
348	4	0.1	2	2.37	395.8	5.1	4,4
348	4	0.1	3	2.01	182.1	4.4	4,4,4
348	4	0.1	4	inf	5.0	-	N.A.
3437	3	∞	2	4.37	287.4	67.5	3,3
3437	3	∞	3	3.5^{*}	3,600	339.8	3,3,3
3437	3	∞	4	3.04^{**}	3,600	133.6	N.A.
3437	3	0.1	2	4.37	323.6	54.2	3,3
3437	3	0.1	3	3.49^{**}	3,600	65.6	N.A.
3437	3	0.1	4	3.05^{**}	3,600	93.8	N.A.
3437	4	∞	2	4.52^{**}	3,600	102.3	N.A.
3437	4	∞	3	3.84^{**}	3,600	7.6	N.A.
3437	4	∞	4	3.75^{**}	3,600	29.1	N.A.
3437	4	0.1	2	4.72	353.5	48.4	4,4
3437	4	0.1	3	inf	50.1	-	N.A.
3437	4	0.1	4	inf	44.3	-	N.A.
hospital-ward-proximity	3	∞	2	1.13	1.9	1.0	5,3
hospital-ward-proximity	3	∞	3	1.00	4.1	1.1	4,5,3
hospital-ward-proximity	3	∞	4	0.95	0.6	0.2	3,4,3,3
hospital-ward-proximity	3	0.1	2	1.13	1.8	0.5	4,3
hospital-ward-proximity	3	0.1	3	1.00	3.2	0.3	3,3,3
hospital-ward-proximity	3	0.1	4	0.95	0.9	0.1	3,3,3,3
hospital-ward-proximity	4	∞	2	1.13	2.5	1.2	4,4
hospital-ward-proximity	4	∞	3	1.00	3.9	0.4	4,4,4
hospital-ward-proximity	4	∞	4	0.95	4.4	2.8	4,4,4,4
hospital-ward-proximity	4	0.1	2	1.13	2.1	0.5	4,4
hospital-ward-proximity	4	0.1	3	1.00	4.1	0.5	4,4,4
hospital-ward-proximity	4	0.1	4	0.95	1.6	0.2	4,4,4,4
ia-workplace-contacts	3	∞	2	1.73	0.8	0.3	3,3
ia-workplace-contacts	3	∞	3	1.29	1.6	1.0	3,3,3
ia-workplace-contacts	3	∞	4	1.18	1.2	0.4	3,3,3,3
ia-workplace-contacts	3	0.1	2	1.76	1.8	0.4	3,3
ia-workplace-contacts	3	0.1	3	1.36	2.0	0.3	3,3,3
ia-workplace-contacts	3	0.1	4	1.18	5.6	1.1	3,3,3,3

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
ia-workplace-contacts	4	∞	2	1.86	2.9	0.8	4,4
ia-workplace-contacts	4	∞	3	1.52	3.9	0.5	4,4,4
ia-workplace-contacts	4	∞	4	1.41	5.4	0.7	5,4,4,4
ia-workplace-contacts	4	0.1	2	1.86	2.7	0.5	4,4
ia-workplace-contacts	4	0.1	3	1.55	4.4	0.3	4,4,4
ia-workplace-contacts	4	0.1	4	1.64	112.5	0.8	4,4,4,4
infect-hyper	3	∞	2	1.06	2.5	1.2	4,3
infect-hyper	3	∞	3	0.97	2.5	0.9	3,3,3
infect-hyper	3	∞	4	0.96	1.9	0.3	3,5,3,3
infect-hyper	3	0.1	2	1.06	17.3	0.9	3,6
infect-hyper	3	0.1	3	0.97	10.8	0.6	3,3,4
infect-hyper	3	0.1	4	0.96	8.9	0.2	3,3,4,4
infect-hyper	4	∞	2	1.06	4.2	1.7	4,4
infect-hyper	4	∞	3	0.97	3.1	1.1	4,4,4
infect-hyper	4	∞	4	0.96	2.5	0.5	4,4,4,4
infect-hyper	4	0.1	2	1.06	11.1	1.0	4,8
infect-hyper	4	0.1	3	0.97	4.4	0.3	4,4,4
infect-hyper	4	0.1	4	0.96	4.5	0.2	4,4,4,4
ia-radoslaw-email	3	∞	2	0.99	6.2	3.0	3,3
ia-radoslaw-email	3	∞	3	0.98	6.2	2.3	3,3,3
ia-radoslaw-email	3	∞	4	0.98	4.8	1.4	3,5,3,3
ia-radoslaw-email	3	0.1	2	0.99	9.9	3.0	3,3
ia-radoslaw-email	3	0.1	3	0.98	19.7	1.8	3,3,3
ia-radoslaw-email	3	0.1	4	0.98	20.1	1.9	3,3,3,3
ia-radoslaw-email	4	∞	2	1.06	25.5	6.8	4,4
ia-radoslaw-email	4	∞	3	1.05	25.4	4.0	4,4,4
ia-radoslaw-email	4	∞	4	1.05	26.6	3.4	4,4,4,5
ia-radoslaw-email	4	0.1	2	1.08	24.8	3.3	4,5
ia-radoslaw-email	4	0.1	3	inf	3.1	-	N.A.
ia-radoslaw-email	4	0.1	4	inf	3.0	-	N.A.
infect-dublin	3	∞	2	3.84	79.0	21.7	3,3
infect-dublin	3	∞	3	3.22	66.2	18.7	3,3,3
infect-dublin	3	∞	4	2.91	85.1	23.9	3,3,3,3
infect-dublin	3	0.1	2	4.03	270.8	17.3	3,3
infect-dublin	3	0.1	3	3.22	65.2	7.3	3,3,3
infect-dublin	3	0.1	4	2.94	405.9	11.0	3,3,3,3
infect-dublin	4	∞	2	4.59^{**}	3,600	68.2	N.A.
infect-dublin	4	∞	3	4.01^{**}	3,600	43.4	N.A.
infect-dublin	4	∞	4	3.52	2,243.1	17.7	4,4,4,4
infect-dublin	4	0.1	2	5.16^{**}	3,600	141.6	N.A.
infect-dublin	4	0.1	3	4.52^{**}	3,600	30.9	N.A.
infect-dublin	4	0.1	4	inf	59.7	-	N.A.
insecta-ant-colony6-day01	3	∞	2	0.99	39.3	22.2	65,3

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
insecta-ant-colony6-day01	3	∞	3	0.98	26.4	9.0	9,76,30
insecta-ant-colony6-day01	3	∞	4	0.98	24.5	7.8	8,4,74,30
insecta-ant-colony6-day01	3	0.1	2	0.99	34.2	10.1	40,28
insecta-ant-colony6-day01	3	0.1	3	0.98	40.3	4.2	41,15,22
insecta-ant-colony6-day01	3	0.1	4	0.98	20.7	1.8	16,13,36,31
insecta-ant-colony6-day01	4	∞	2	0.99	27.3	8.0	18,44
insecta-ant-colony6-day01	4	∞	3	0.98	22.1	2.7	12,29,26
insecta-ant-colony6-day01	4	∞	4	0.98	21.0	1.6	9,17,15,31
insecta-ant-colony6-day01	4	0.1	2	0.99	36.9	10.6	34,44
insecta-ant-colony6-day01	4	0.1	3	0.98	30.1	3.9	37,18,22
insecta-ant-colony6-day01	4	0.1	4	0.98	23.3	2.0	29,23,9,16
aves-wildbird-network	3	∞	2	1.57	47.1	27.0	3,3
aves-wildbird-network	3	∞	3	1.36	26.0	10.6	4,3,3
aves-wildbird-network	3	∞	4	1.14	28.6	8.8	3,3,3,3
aves-wildbird-network	3	0.1	2	1.61	43.1	11.6	6,3
aves-wildbird-network	3	0.1	3	1.36	39.9	11.4	3,3,3
aves-wildbird-network	3	0.1	4	1.14	31.5	4.2	3,3,3,3
aves-wildbird-network	4	∞	2	1.60	133.2	13.6	6,4
aves-wildbird-network	4	∞	3	1.42	217.7	28.8	4,4,4
aves-wildbird-network	4	∞	4	1.31	124.8	36.4	4,4,4,4
aves-wildbird-network	4	0.1	2	1.64	140.1	11.7	5,4
aves-wildbird-network	4	0.1	3	1.42	146.0	11.7	4,4,4
aves-wildbird-network	4	0.1	4	1.31	235.1	13.9	4,4,4,4
mammalia-dolphin-florida-overall	3	∞	2	2.47	186.4	28.0	3,3
mammalia-dolphin-florida-overall	3	∞	3	2.09	77.7	16.1	3,3,3
mammalia-dolphin-florida-overall	3	∞	4	1.87	188.0	35.4	3,3,3,3
mammalia-dolphin-florida-overall	3	0.1	2	2.47	100.5	39.9	3,3
mammalia-dolphin-florida-overall	3	0.1	3	2.09	128.6	17.0	3,3,3
mammalia-dolphin-florida-overall	3	0.1	4	1.89	351.2	30.7	3,3,3,3
mammalia-dolphin-florida-overall	4	∞	2	2.71	1,014.3	23.9	4,4
mammalia-dolphin-florida-overall	4	∞	3	2.41^{**}	3,600	66.6	N.A.
mammalia-dolphin-florida-overall	4	∞	4	2.35	1,779.6	334.1	4,4,4,4
mammalia-dolphin-florida-overall	4	0.1	2	2.71	1,211.7	28.3	4,4
mammalia-dolphin-florida-overall	4	0.1	3	2.65^{**}	3,600	27.0	N.A.
mammalia-dolphin-florida-overall	4	0.1	4	2.81^{**}	3,600	-	N.A.
mycielskian7	3	∞	2	1.74	15.2	0.9	3,3
mycielskian7	3	∞	3	1.44	107.4	0.6	4,3,3
mycielskian7	3	∞	4	1.40	1,977.6	0.6	3,3,3,3
mycielskian7	3	0.1	2	1.74	26.2	0.6	3,3
mycielskian7	3	0.1	3	1.47	107.7	0.4	3,3,3
mycielskian7	3	0.1	4	1.46	2,768.8	0.5	3,3,3,3
mycielskian7	4	∞	2	1.77	30.6	0.5	4,4
mycielskian7	4	∞	3	1.57	196.9	0.5	4,4,4

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
mycielskian7	4	∞	4	1.62*	3,600	0.6	4,4,4,4
mycielskian7	4	0.1	2	1.77	36.5	0.5	4,4
mycielskian7	4	0.1	3	1.76	748.0	0.6	4,4,4
mycielskian7	4	0.1	4	1.81	3,221.2	0.3	4,4,4,4
mycielskian8	3	∞	2	1.74	55.7	5.1	3,3
mycielskian8	3	∞	3	1.41	395.4	1.5	3,3,3
mycielskian8	3	∞	4	1.17*	3,600	2.7	3,3,3,3
mycielskian8	3	0.1	2	1.74	232.2	3.7	3,3
mycielskian8	3	0.1	3	1.41	788.5	1.2	3,3,3
mycielskian8	3	0.1	4	1.17*	3,600	1.3	3,3,3,3
mycielskian8	4	∞	2	1.74	389.3	3.3	4,4
mycielskian8	4	∞	3	1.42	810.4	0.8	4,4,4
mycielskian8	4	∞	4	1.28*	3,600	1.4	4,4,4,4
mycielskian8	4	0.1	2	1.74	171.1	5.4	4,4
mycielskian8	4	0.1	3	1.44*	3,600	2.5	4,4,4
mycielskian8	4	0.1	4	1.28*	3,600	2.2	4,4,4,4
mycielskian9	3	∞	2	1.75	215.4	18.0	4,3
mycielskian9	3	∞	3	1.39	2,737.3	12.6	3,3,3
mycielskian9	3	∞	4	1.09*	3,600	5.2	3,3,3,3
mycielskian9	3	0.1	2	1.75	601.4	28.5	3,4
mycielskian9	3	0.1	3	1.2*	3,600	10.3	3,3,3
mycielskian9	3	0.1	4	1.1*	3,600	8.6	3,3,3,3
mycielskian9	4	∞	2	1.75	1,751.8	23.0	4,4
mycielskian9	4	∞	3	1.19*	3,600	13.8	4,4,4
mycielskian9	4	∞	4	1.14*	3,600	7.2	4,4,4,4
mycielskian9	4	0.1	2	1.75	985.5	23.1	4,4
mycielskian9	4	0.1	3	1.2*	3,600	8.8	4,4,4
mycielskian9	4	0.1	4	1.15*	3,600	8.3	4,4,4,4
qc324	3	∞	2	2.46*	3,600	45.1	38,43
qc324	3	∞	3	1.7*	3,600	34.2	80,19,62
qc324	3	∞	4	0.99	80.9	54.0	80,80,80,80
qc324	3	0.1	2	2.47	2,936.5	66.5	16,65
qc324	3	0.1	3	1.7*	3,600	24.8	17,38,37
qc324	3	0.1	4	0.99	113.2	82.0	80,80,80,80
qc324	4	∞	2	2.46*	3,600	92.4	47,34
qc324	4	∞	3	1.7*	3,600	49.4	80,37,44
qc324	4	∞	4	0.99	110.0	73.4	80,80,80,80
qc324	4	0.1	2	2.46*	3,600	137.2	65,16
qc324	4	0.1	3	1.7*	3,600	36.1	17,42,27
qc324	4	0.1	4	0.99	76.0	41.2	80,80,80,80
ex2	3	∞	2	2.33	75.4	43.5	10,5
ex2	3	∞	3	1.85	262.9	184.3	4,6,3
ex2	3	∞	4	1.51	80.0	40.0	14,5,10,14

Table D.4: Results for HCGP with $Q \in \{3, 4\}$ (continued).

Graph (G)	Q	τ	K	OPT	OPT time (s)	Callback time (s)	Conn.
ex2	3	0.1	2	2.33	81.0	44.0	10,9
ex2	3	0.1	3	1.85	193.2	139.9	6,3,5
ex2	3	0.1	4	1.51	68.3	16.9	12,8,10,12
ex2	4	∞	2	2.33	72.5	28.5	10,8
ex2	4	∞	3	1.85	212.7	148.8	4,6,4
ex2	4	∞	4	1.51	116.9	67.4	19,6,5,19
ex2	4	0.1	2	2.33	88.6	47.5	5,10
ex2	4	0.1	3	1.85	187.9	127.1	4,6,4
ex2	4	0.1	4	1.51	83.6	21.9	12,9,8,14

Table D.5: Results for HCGP with $Q = 1$.

Graph (G)	τ	K	OPT	OPT time (s)
Sioux Falls	∞	2	2.54	0.1
Sioux Falls	∞	3	1.46	0.0
Sioux Falls	∞	4	1.00	0.1
Sioux Falls	0.1	2	2.54	0.1
Sioux Falls	0.1	3	1.54	0.1
Sioux Falls	0.1	4	1.04	0.1
Berlin-Friedrichshain	∞	2	21.08	1.4
Berlin-Friedrichshain	∞	3	14.68	2.1
Berlin-Friedrichshain	∞	4	9.96	1.9
Berlin-Friedrichshain	0.1	2	21.08	11.0
Berlin-Friedrichshain	0.1	3	14.94	5.3
Berlin-Friedrichshain	0.1	4	9.96	5.9
Berlin-Prenzlauerberg-Center	∞	2	28.41	4.6
Berlin-Prenzlauerberg-Center	∞	3	17.80	4.6
Berlin-Prenzlauerberg-Center	∞	4	12.47	4.1
Berlin-Prenzlauerberg-Center	0.1	2	28.41	36.8
Berlin-Prenzlauerberg-Center	0.1	3	17.91	35.5
Berlin-Prenzlauerberg-Center	0.1	4	12.57	23.9
Berlin-Tiergarten	∞	2	33.69	13.6
Berlin-Tiergarten	∞	3	22.04	8.2
Berlin-Tiergarten	∞	4	17.17	8.1
Berlin-Tiergarten	0.1	2	33.69	126.0
Berlin-Tiergarten	0.1	3	25.18	279.2
Berlin-Tiergarten	0.1	4	18.70	116.1
Berlin-Mitte-Center	∞	2	24.87	6.3
Berlin-Mitte-Center	∞	3	17.54	10.3
Berlin-Mitte-Center	∞	4	12.16	5.6

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
Berlin-Mitte-Center	0.1	2	25.15	58.7
Berlin-Mitte-Center	0.1	3	18.08	63.1
Berlin-Mitte-Center	0.1	4	12.57	57.9
Anaheim	∞	2	27.44	10.3
Anaheim	∞	3	18.69	18.6
Anaheim	∞	4	13.21	8.0
Anaheim	0.1	2	27.44	160.8
Anaheim	0.1	3	18.98	90.7
Anaheim	0.1	4	13.51	47.0
Barcelona	∞	2	56.70	179.8
Barcelona	∞	3	36.73	277.9
Barcelona	∞	4	28.57	477.6
Barcelona	0.1	2	56.92	285.5
Barcelona	0.1	3	36.73	344.1
Barcelona	0.1	4	29.15	969.4
Chicago-Sketch	∞	2	43.30	38.2
Chicago-Sketch	∞	3	28.66	46.6
Chicago-Sketch	∞	4	22.17	58.4
Chicago-Sketch	0.1	2	43.30	36.9
Chicago-Sketch	0.1	3	28.99	79.8
Chicago-Sketch	0.1	4	22.17	74.3
Terrassa	∞	2	84.00	191.5
Terrassa	∞	3	53.54	240.1
Terrassa	∞	4	39.67	3,291.4
Terrassa	0.1	2	84.00	212.6
Terrassa	0.1	3	0.0^{**}	3,600
Terrassa	0.1	4	39.42^{**}	3,600
Rhode Island	∞	2	13.05	7.9
Rhode Island	∞	3	8.39	6.5
Rhode Island	∞	4	6.21	6.5
Rhode Island	0.1	2	13.07	22.7
Rhode Island	0.1	3	8.64	49.1
Rhode Island	0.1	4	6.21	11.9
New Hampshire	∞	2	20.18	16.8
New Hampshire	∞	3	12.93	18.4
New Hampshire	∞	4	9.97	18.3
New Hampshire	0.1	2	20.42	446.1
New Hampshire	0.1	3	13.01	138.9
New Hampshire	0.1	4	9.97	797.8
Maine	∞	2	17.36	19.7
Maine	∞	3	13.30	55.1
Maine	∞	4	10.16	49.1
Maine	0.1	2	17.63	350.3

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
Maine	0.1	3	13.95	2,118.5
Maine	0.1	4	10.22	609.1
Idaho	∞	2	19.95	26.8
Idaho	∞	3	12.93	35.9
Idaho	∞	4	9.82	31.9
Idaho	0.1	2	20.96	347.0
Idaho	0.1	3	14.48	281.7
Idaho	0.1	4	10.20	118.7
West Virginia	∞	2	25.31	29.3
West Virginia	∞	3	16.90	34.1
West Virginia	∞	4	11.84	30.4
West Virginia	0.1	2	25.37	412.3
West Virginia	0.1	3	16.91	104.8
West Virginia	0.1	4	12.05	581.5
Nebraska	∞	2	22.06	40.3
Nebraska	∞	3	14.78	55.0
Nebraska	∞	4	11.57	50.6
Nebraska	0.1	2	26.33	807.9
Nebraska	0.1	3	15.60	486.9
Nebraska	0.1	4	11.89	383.4
New Mexico	∞	2	17.25	65.6
New Mexico	∞	3	12.92	90.0
New Mexico	∞	4	11.13	72.3
New Mexico	0.1	2	17.26	517.8
New Mexico	0.1	3	13.00	820.8
New Mexico	0.1	4	12.00	1,208.3
Utah	∞	2	17.50	104.7
Utah	∞	3	14.50	214.4
Utah	∞	4	11.73	261.0
Utah	0.1	2	17.50	884.3
Utah	0.1	3	0.0^{**}	3,600
Utah	0.1	4	12.9^{**}	3,600
Arkansas	∞	2	35.28	124.3
Arkansas	∞	3	23.86	204.7
Arkansas	∞	4	17.81	212.8
Arkansas	0.1	2	0.0^{**}	3,600
Arkansas	0.1	3	24.02	3,141.7
Arkansas	0.1	4	0.0^{**}	3,600
Nevada	∞	2	23.23	147.6
Nevada	∞	3	16.15	133.1
Nevada	∞	4	13.45	296.7
Nevada	0.1	2	0.0^{**}	3,600
Nevada	0.1	3	17.79	2,101.5

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
Nevada	0.1	4	13.76	1,781.9
Mississippi	∞	2	39.38	134.3
Mississippi	∞	3	25.23	201.0
Mississippi	∞	4	20.85	966.4
Mississippi	0.1	2	39.38	3,175.9
Mississippi	0.1	3	0.0**	3,600
Mississippi	0.1	4	0.0**	3,600
Iowa	∞	2	33.21	147.5
Iowa	∞	3	24.38	195.5
Iowa	∞	4	19.54	339.6
Iowa	0.1	2	0.0**	3,600
Iowa	0.1	3	0.0**	3,600
Iowa	0.1	4	0.0**	3,600
IEEE300Bus	∞	2	5.12	2.3
IEEE300Bus	∞	3	4.55	2.7
IEEE300Bus	∞	4	4.22	2.9
IEEE300Bus	0.1	2	7.65	43.0
IEEE300Bus	0.1	3	7.98	388.6
IEEE300Bus	0.1	4	7.53**	3,600
power-494-bus	∞	2	6.24	12.8
power-494-bus	∞	3	5.74	14.0
power-494-bus	∞	4	5.29	13.4
power-494-bus	0.1	2	8.2**	3,600
power-494-bus	0.1	3	8.62**	3,600
power-494-bus	0.1	4	8.84**	3,600
3980	∞	2	1.42	0.1
3980	∞	3	1.12	0.2
3980	∞	4	0.98	0.3
3980	0.1	2	1.42	0.2
3980	0.1	3	1.29	0.8
3980	0.1	4	1.10	0.4
698	∞	2	1.75	0.2
698	∞	3	1.44	0.3
698	∞	4	1.18	0.4
698	0.1	2	1.75	0.2
698	0.1	3	1.54	1.2
698	0.1	4	1.33	0.5
414	∞	2	1.89	1.8
414	∞	3	1.42	3.1
414	∞	4	1.25	4.3
414	0.1	2	2.21	4.5
414	0.1	3	1.66	4.7
414	0.1	4	1.41	26.6

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
686	∞	2	2.17	2.6
686	∞	3	1.85	5.3
686	∞	4	1.53	3.1
686	0.1	2	2.17	13.2
686	0.1	3	1.85	22.0
686	0.1	4	1.64	18.2
348	∞	2	2.25	8.0
348	∞	3	1.76	7.1
348	∞	4	1.58	13.2
348	0.1	2	2.25	110.8
348	0.1	3	1.76	75.5
348	0.1	4	1.58	98.9
3437	∞	2	4.34	131.3
3437	∞	3	3.37	143.4
3437	∞	4	2.83	94.8
3437	0.1	2	4.34	204.5
3437	0.1	3	3.37	3,095.6
3437	0.1	4	2.84	443.9
hospital-ward-proximity	∞	2	1.13	0.7
hospital-ward-proximity	∞	3	0.96	0.5
hospital-ward-proximity	∞	4	0.95	0.5
hospital-ward-proximity	0.1	2	1.13	0.8
hospital-ward-proximity	0.1	3	1.00	2.1
hospital-ward-proximity	0.1	4	0.95	0.7
ia-workplace-contacts	∞	2	1.70	0.5
ia-workplace-contacts	∞	3	1.26	0.5
ia-workplace-contacts	∞	4	1.12	0.6
ia-workplace-contacts	0.1	2	1.70	0.7
ia-workplace-contacts	0.1	3	1.26	0.7
ia-workplace-contacts	0.1	4	1.12	0.7
infect-hyper	∞	2	1.06	1.4
infect-hyper	∞	3	0.97	1.5
infect-hyper	∞	4	0.96	1.4
infect-hyper	0.1	2	1.06	1.9
infect-hyper	0.1	3	0.97	3.7
infect-hyper	0.1	4	0.96	1.8
ia-radoslaw-email	∞	2	0.99	2.6
ia-radoslaw-email	∞	3	0.98	2.5
ia-radoslaw-email	∞	4	0.98	2.7
ia-radoslaw-email	0.1	2	0.99	3.0
ia-radoslaw-email	0.1	3	0.98	3.4
ia-radoslaw-email	0.1	4	0.98	3.2
infect-dublin	∞	2	3.75	20.9

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
infect-dublin	∞	3	3.17	22.5
infect-dublin	∞	4	2.82	27.8
infect-dublin	0.1	2	3.75	56.6
infect-dublin	0.1	3	3.17	45.0
infect-dublin	0.1	4	2.82	57.1
insecta-ant-colony6-day01	∞	2	0.99	7.0
insecta-ant-colony6-day01	∞	3	0.98	6.8
insecta-ant-colony6-day01	∞	4	0.98	6.8
insecta-ant-colony6-day01	0.1	2	0.99	9.2
insecta-ant-colony6-day01	0.1	3	0.98	8.9
insecta-ant-colony6-day01	0.1	4	0.98	7.4
aves-wildbird-network	∞	2	1.55	10.2
aves-wildbird-network	∞	3	1.34	6.1
aves-wildbird-network	∞	4	1.14	7.2
aves-wildbird-network	0.1	2	1.60	19.8
aves-wildbird-network	0.1	3	1.34	27.4
aves-wildbird-network	0.1	4	1.14	29.3
mammalia-dolphin-florida-overall	∞	2	2.46	12.2
mammalia-dolphin-florida-overall	∞	3	2.06	12.7
mammalia-dolphin-florida-overall	∞	4	1.83	18.6
mammalia-dolphin-florida-overall	0.1	2	2.46	21.5
mammalia-dolphin-florida-overall	0.1	3	2.06	36.7
mammalia-dolphin-florida-overall	0.1	4	1.83	85.1
mycielskian7	∞	2	1.71	2.1
mycielskian7	∞	3	1.32	16.1
mycielskian7	∞	4	1.12	18.2
mycielskian7	0.1	2	1.71	4.2
mycielskian7	0.1	3	1.32	16.9
mycielskian7	0.1	4	1.12	32.3
mycielskian8	∞	2	1.73	29.1
mycielskian8	∞	3	1.35	66.3
mycielskian8	∞	4	1.15	136.4
mycielskian8	0.1	2	1.73	63.7
mycielskian8	0.1	3	1.35	181.9
mycielskian8	0.1	4	1.15	253.8
mycielskian9	∞	2	1.74	729.9
mycielskian9	∞	3	1.19^*	3,600
mycielskian9	∞	4	1.17	658.3
mycielskian9	0.1	2	1.74	138.2
mycielskian9	0.1	3	1.17^*	3,600
mycielskian9	0.1	4	1.03^*	3,600
qc324	∞	2	2.47	309.1
qc324	∞	3	1.7^*	3,600

Table D.5: Results for HCGP with $Q = 1$ (continued).

Graph (G)	τ	K	OPT	OPT time (s)
qc324	∞	4	0.99	37.2
qc324	0.1	2	2.47	352.5
qc324	0.1	3	1.7*	3,600
qc324	0.1	4	0.99	48.5
ex2	∞	2	2.33	35.9
ex2	∞	3	1.85	66.3
ex2	∞	4	1.51	32.5
ex2	0.1	2	2.33	31.9
ex2	0.1	3	1.85	35.8
ex2	0.1	4	1.51	75.0