# A guide to inexact contiguity constraints

Ishaan Jolly Austin Buchanan

August 17, 2026

**Abstract**

For decades, researchers have complained that contiguity constraints make problems like districting much harder than other combinatorial optimization problems. This has prompted the use of integer programming models with *inexact* contiguity constraints that may be fast in practice but are invalid in the sense that they cut off or "overlook" some contiguous solutions. Examples include the tree-based constraints of Zoltners and Sinha (*Management Science*, 1983), the distance-based constraints of Mehrotra et al. (*Management Science*, 1998), and the DAG-based constraints of Shahmizad and Buchanan (*Operations Research*, 2025). In this paper, we compare these models for imposing contiguity, both analytically and computationally. We identify their sizes, the number of solutions that they admit, the precise conditions under which they are integral, and the computational complexity of optimizing over them. For example, we can optimize a linear objective over the tree-based constraints in polynomial time, but the same problem is NP-hard for distance-based constraints. To reveal the strengths and weaknesses of these models in practice, we use them to draw political districts at different levels of granularity—counties ($n \approx 100$ vertices), voting precincts ($n \approx 2{,}500$), and census blocks ($n \approx 175{,}000$)—also comparing with exact methods. The experiments reveal some surprising phenomena, e.g., on the largest graphs, the largest model (DAG) not only produces the best solutions, but also solves quickest. Finally, we report how many enacted congressional and legislative districts across the USA satisfy the various inexact contiguity constraints. The result is a guide to the effectiveness and computational limits of each approach and a codebase for future researchers.

**Keywords:** integer programming; connectivity; contiguity; tree; DAG; political district;

## 1 Introduction

For decades, researchers have complained that contiguity constraints make problems like political districting much harder than other combinatorial optimization problems [17, 27, 44, 45, 50]. This has prompted the use of integer programming models with *inexact* contiguity constraints that may be fast in practice but are invalid in the sense that they cut off or "overlook" some contiguous solutions. Prominent examples include the tree-based constraints of Zoltners and Sinha [62] and the distance-based constraints of Mehrotra et al. [38], see also Cova and Church [17]. Recently, Shahmizad and Buchanan [47] proposed a directed acyclic graph (DAG) generalization.

In each of the aforementioned papers, the authors tout the benefits of their own approach but do not compare with previous ones, providing little guidance for future researchers and

practitioners as to which approach will best suit their needs. Moreover, the speeds of computer hardware and MIP software have increased substantially in recent decades [1, 8, 26]—not to mention recent advances in modeling contiguity [12, 22, 56]. As a result, computational experiments from these older papers are now wildly outdated and provide little insight to how these models would perform today. Our intent in the present paper is to help fill this gap and inform researchers and practitioners to the effectiveness and computational limits of each approach, analyzing each model and empirically testing them on synthetic and real-life instances.

To formalize the discussion, consider a simple graph $G = (V, E)$ with vertex set $V$ and edge set $E \subseteq \binom{V}{2}$, where $\binom{V}{2} := \{S \subseteq V : |S| = 2\}$. We denote the number of vertices and edges by $n = |V|$ and $m = |E|$, respectively. Each vertex of $G$ represents a geographic unit, such as a county, census tract, voting precinct, or census block. Edges indicate which pairs of geographic units are adjacent on the map, i.e., share a border of positive length. Figure 1 shows Iowa's counties and the associated county-level graph with 99 vertices and 222 edges.

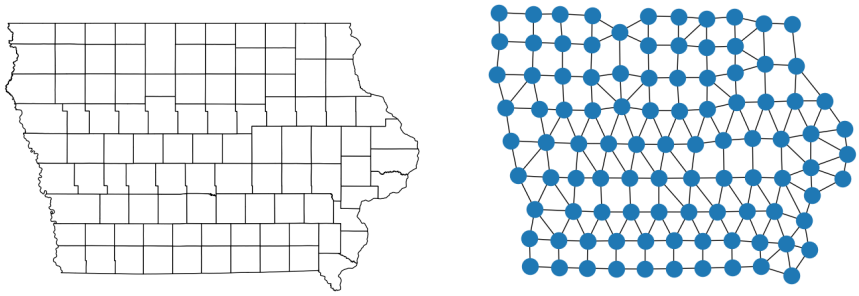

Figure 1: Iowa's 99 counties as a graph

In the USA, each state is divided into counties, each county is divided into census tracts (and voting precincts), and each census tract is divided into census blocks. To give an idea of the scale involved, Iowa has 99 counties, 896 census tracts, 2,536 voting precincts, and 175,199 census blocks. The associated tract-level, precinct-level, and block-level graphs are too large to visualize and are thus not shown, but we will run experiments on them later.

For simplicity, suppose we want to construct a single district $D \subseteq V$ that is contiguous, i.e., its induced subgraph $G[D] := (D, E \cap \binom{D}{2})$ is connected. In an integer programming model, introduce the variables:

$$x_i = \begin{cases} 1 & \text{if vertex } i \in V \text{ is chosen in the district } D \\ 0 & \text{if otherwise.} \end{cases}$$

Below, we recall the tree-based, distance-based, and DAG-based constraints for contiguity.

The tree-based constraints of Zoltners and Sinha [62] can be described as follows. Pick a root vertex $r \in V$ and compute a shortest path tree rooted at $r$. In this tree, each vertex $i \in V \setminus \{r\}$ will have one incoming edge, and the tail of this edge is called the *predecessor* of $i$, denoted $\text{pred}(i)$. The tree-based constraints are then:

$$x_i \leq x_{\text{pred}(i)} \qquad \forall i \in V \setminus \{r\}. \tag{1}$$

Zoltners and Sinha originally proposed to compute a shortest path tree in an edge-weighted graph, with edge weights based on travel times.

The distance-based contiguity constraints of Mehrotra et al. [38] can be described as follows, see also Cova and Church [17]. Pick a root vertex $r \in V$ and compute the distance $\text{dist}(j)$ to every vertex $j \in V$. Denote the neighborhood of vertex $i$ in $G$ by

$$N(i) \coloneqq \{j \in V \mid \{i, j\} \in E\}.$$

If using hop-based distances, the distance-based constraints are then:

$$x_i \leq \sum_{j \in N(i) \,:\, \text{dist}(j) = \text{dist}(i) - 1} x_j \qquad \forall i \in V \setminus \{r\}. \tag{2}$$

Mehrotra et al. and Cova and Church both used unweighted or *hop-based* distances, but the same idea can be extended to edge-weighted distances.

The DAG-based constraints of Shahmizad and Buchanan [47] are as follows. Orient each undirected edge $\{i, j\} \in E$ as either $(i, j)$ or $(j, i)$ so that the resulting directed graph $H = (V, A)$ has no directed cycles. Denote the in-neighborhood of vertex $i$ in digraph $H$ by

$$N^-(i) \coloneqq \{j \in V \mid (j, i) \in A\}.$$

The DAG-based constraints are then:

$$x_i \leq \sum_{j \in N^-(i)} x_j \qquad \forall i \in V \setminus \{r\}. \tag{3}$$

Shahmizad and Buchanan obtained an orientation of $G$ as follows. Pick a root vertex $r \in V$ and compute distances to all others. Sort the vertices by distance and orient the edge $\{i, j\}$ as $(i, j)$ if $i$ comes before $j$ in the sorted order.

By now, it may be clear that each model can be defined using a directed acyclic graph $H = (V, A)$ just as in (3) but with respect to different directed edge sets $A$. Figure 2 illustrates these directed graphs when applied to Iowa's county-level graph. The tree-based, distance-based, and DAG-based constraints use 98, 161, and 221 edges, respectively. Generally, the tree-based constraints use the bare minimum of $n - 1$ edges, the DAG-based constraints use all $m$ edges, and the distance-based constraints use some number in between.

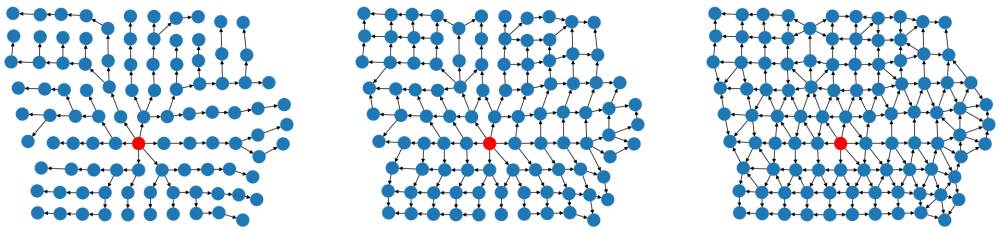

Figure 2: Visualizing the tree, distance, and DAG constraints

The number of contiguous solutions allowed by the constraints (3) will generally increase as we add more edges. This is the primary reason for moving from tree-based to distance-based to DAG-based constraints. However, this may come at a computational cost, hence

the desire to better understand the possible tradeoffs between them in terms of solution quality and computational speed. We note that the tree-based constraints already permit 3,620,266,575,184,897$\sim 10^{15}$ contiguous districts when applied to Iowa's 99 counties using Polk County as root, as we will see in equations (10) and (11). Meanwhile, if using precincts and blocks, the numbers grow to the order of $10^{385}$ and $10^{32,078}$, respectively.

**Our contributions.** We compare the tree-based, distance-based, and DAG-based models for imposing contiguity, both analytically and computationally. We identify the sizes of these models (Table 1), the number of solutions that they admit (Theorem 1 and equations (10) and (11)), the precise conditions under which they are integral (Theorem 2), and the computational complexity of optimizing over them (Theorems 3 and 4 and Proposition 2). To reveal their strengths and weaknesses in practice, we use them to draw districts for Iowa at different scales: counties ($n = 99$), voting precincts ($n = 2{,}536$), and census blocks ($n = 175{,}199$). We also compare with exact methods. The experiments reveal some surprising and nuanced phenomena. Finally, we report how many enacted congressional and legislative districts across the USA satisfy the various inexact contiguity constraints. For example, we find that only 2.27% of enacted districts satisfy the tree-based constraints (under hop-based distances), but this improves to 28.74% if using DAG-based constraints (under a particular edge-weighting scheme). Our code, instances, and results are publicly available at: `https://github.com/Ishaanjolly/A-guide-to-inexact-contiguity-constraints/`

**Outline.** The paper is organized as follows. Section 2 provides background information about exact and inexact integer programming models for imposing contiguity and surveys the relevant literature. Section 3 analytically studies the inexact models, including their size, integrality, and complexity. Section 4 computationally compares and evaluates them on synthetic and real-life districting instances. We conclude in Section 5.

# 2 Background and Literature Review

Connectivity or contiguity constraints arise in all sorts of problems, including political districting [9, 25, 31, 38, 56], sales districting [20, 30, 62], virtual backbone design in wireless networks [10, 23, 54], cluster detection in bioinformatics [4, 5, 18], forest harvesting and conservation planning [12, 33, 40], logistics districting and fulfillment regionalization [11, 29, 34], and image segmentation [13, 57]. Unfortunately, despite all these applications, researchers have frequently complained about the challenges of imposing contiguity constraints in a computationally tractable way. In a survey paper on districting, Ricca et al. [44] state that "[contiguity] is particularly difficult to deal with and, sometimes, it is even discarded from [political districting] models and considered only a posteriori". Often, researchers hope that a compactness objective will produce (nearly-)contiguous solutions as a byproduct [6, 31, 44].

## 2.1 Inexact Contiguity Constraints

These difficulties motivated the tree-based constraints of Zoltners and Sinha [62] in 1983. It should be noted that Zoltners and Sinha explicitly considered the use of additional edges beyond those from the shortest path tree, writing that:

> *Additional edges can be derived by obtaining near optimal shortest paths (e.g., the second shortest, third shortest, etc.). However, in most cases additional edges would be supplied by experienced sales managers.*

Even with these simplistic contiguity constraints, Zoltners and Sinha mention that "[t]he computational effort required to solve [their model] using general purpose integer programming algorithms is prohibitive" and instead propose to use Lagrangian relaxation.

In 1998, Mehrotra et al. [38] propose a column generation heuristic for districting where the pricing step involves identifying a district with negative reduced cost. Contiguity of the district is imposed by constraints of the form $x_i \leq \sum_{j \in S_i} x_j$ where

$$S_i = \{j \in N(i) \mid \text{dist}(r, j) = \text{dist}(r, i) - 1\}$$

is the set of neighbors of $i$ that are closer to root $r$ under hop-based distances. Mehrotra et al. state that this is intended to enforce that the district is a subtree of a shortest path tree. Generalizing this to weighted distances with edge weights $w_{ij} = w_{ji}$ for $\{i, j\} \in E$ gives

$$S_i = \{j \in N(i) \mid \text{dist}(r, j) = \text{dist}(r, i) - w_{ji}\}.$$

In 2000, Cova and Church [17] propose essentially the same model, also using hop-based distances. Mehrotra et al. apply their methods to district the state of South Carolina into $k = 6$ districts, essentially at the county level, with 51 vertices. Cova and Church seek a single district in a 30×30 grid graph (900 vertices) and in a realistic graph with 258 vertices.

In 2025, Shahmizad and Buchanan [47] generalized the distance-based constraints to DAG-based constraints, as follows. Calculate edge-weighted distances from a prescribed root vertex and sort the vertices by distance. For every edge $\{i, j\} \in E$, orient it as $(i, j)$ if $i$ comes before $j$ in the sorted order, and as $(j, i)$ otherwise. When working with block-level graphs, Shahmizad and Buchanan weight each edge $e = \{i, j\} \in E$ as follows:

$$w_e = \begin{cases} 1 & \text{if vertices } i, j \in V \text{ belong to the same census tract} \\ |V| & \text{if vertices } i, j \in V \text{ belong to the same county, but different tracts} \\ |V|^2 & \text{if vertices } i, j \in V \text{ belong to different counties.} \end{cases}$$

By this weighting, distances within a county (or tract) will be shorter than across county boundaries, cf. [14], thus promoting the ability of counties (or tracts) to remain intact. They use these constraints to draw districting plans with a minimum number of county splits, applying them to block-level instances where the number of vertices approaches one million. With a problem-specific decomposition procedure, they find provably optimal solutions for all congressional, state senate, and state house instances. However, solve times for their `detail` step are not reported; this is the subroutine where the DAG constraints are used.

## 2.2 Exact Contiguity Constraints

Most *exact* models for imposing contiguity rely on cuts or flows. We review two popular models—Shirabe's flow model and an $a, b$-separator model—in some detail, and then briefly review other alternatives.

### 2.2.1 Shirabe's flow model

In 2005, Shirabe [48] proposed a flow-based model, whose main idea is to send flow from a root vertex to the other vertices in the district. One advantage of this model is its small size, requiring $O(m + n)$ additional variables, constraints, and nonzero coefficients. For planar graphs, this is only $O(n)$. To describe this model, introduce binary variables $y_i$ indicating

whether vertex $i \in V$ is chosen as the root, and continuous variables $f_{ij}$ indicating how much flow is sent across directed edge $(i, j)$. The constraints are then:

$$\sum_{i \in V} y_i = 1 \tag{4a}$$

$$\sum_{j \in N(i)} (f_{ji} - f_{ij}) \geq x_i - M y_i \qquad \forall i \in V \tag{4b}$$

$$\sum_{j \in N(i)} f_{ji} \leq (M-1) x_i \qquad \forall i \in V \tag{4c}$$

$$f_{ij} \geq 0 \qquad \forall i \in V,\ \forall j \in N(i) \tag{4d}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V. \tag{4e}$$

Constraint (4a) ensures that only one vertex is chosen as root. Constraints (4b) ensure that selected non-root vertices consume one unit of flow. The big-$M$ deactivates this constraint when a vertex is chosen as root, as it will need to "produce" the district's flow. The value of $M$ can be set to $|V|$, or a smaller value like $M = \max\{|D| : \sum_{i \in D} p_i \leq U\}$ if the population of the district should be at most $U$, where $p_i$ is the population of vertex $i \in V$, see [55]. Constraints (4c) prevent flow from entering unselected vertices. These constraints can be strengthened to

$$\sum_{j \in N(i)} f_{ji} \leq (M-1)(x_i - y_i) \qquad \forall i \in V,$$

also preventing flow from entering the root [55]. If only one vertex $r \in V$ is permitted as root, then the constraints reduce to:

$$\sum_{j \in N(i)} (f_{ji} - f_{ij}) = x_i \qquad \forall i \in V \setminus \{r\} \tag{5a}$$

$$\sum_{j \in N(i)} f_{ji} \leq (M-1) x_i \qquad \forall i \in V \setminus \{r\} \tag{5b}$$

$$\sum_{j \in N(r)} f_{jr} = 0 \tag{5c}$$

$$f_{ij} \geq 0 \qquad \forall i \in V,\ \forall j \in N(i). \tag{5d}$$

Shirabe [48] applies these single-district models to a realistic instance with 179 vertices, which yields solve times on the order of seconds or minutes, and to a grid graph of $10 \times 10$ (100 vertices), which yields solve times less than one minute. In a subsequent 2009 paper, Shirabe [49] extends the flow-based model to the multi-district case. Experiments are conducted to partition the $|V| = 48$ contiguous US states into four contiguous districts, with or without prescribed roots. Solve times for various problem types ranged from one second for a compactness objective with prescribed roots, to 20 minutes for a compactness objective without prescribed roots, to more than 2 days for a min-deviation objective without prescribed roots. Recent works apply Shirabe's contiguity constraints to districting problems with roughly 100 to 200 vertices [39, 50]. Because of the big-$M$ constraints, numerical stability issues rear their head on larger instances, as experienced by Validi et al. [56].

### 2.2.2 Separator model

In 2013, Carvajal et al. [12] proposed a separator-based model. It requires no additional variables, but has exponentially many constraints, which are typically implemented in a branch-and-cut fashion. Let $a, b \in V$ be nonadjacent vertices and let $S \subseteq V \setminus \{a, b\}$ be an $a, b$-separator; this means that there is no path between $a$ and $b$ in $G - S$. (Here, $G - S$ is the graph obtained from $G$ by removing vertices $S$ and any incident edges.) Ideally, $S$ is (inclusion-wise) *minimal*, meaning that no proper subset of $S$ is an $a, b$-separator. The associated inequality

$$(a, b\text{-separator inequality, for single district}) \qquad x_a + x_b \leq 1 + \sum_{i \in S} x_i \tag{6}$$

ensures that $a$ and $b$ cannot be selected simultaneously if none of the separator vertices are selected. There are exponentially many of these inequalities, but they can be separated efficiently, in fact in linear time *if* the point $x^*$ to separate is integral, see Algorithm 1 of [22], which also ensures minimality of the separator. The resulting inequalities are strong—the minimal $a, b$-separator inequalities induce facets of the associated connected subgraph polytope when the input graph is connected [58]. It has been observed that few of these inequalities are needed in practice, especially if using a compactness objective that produces nearly contiguous districts [56]. Although practitioners may be wary of implementing $a, b$-separator inequalities given the need to use cut callbacks, just 20 lines of Python code suffice, making separator-based and flow-based models similarly difficult to implement. Further, separator constraints are less prone to numerical issues due to the lack of big-$M$ coefficients.

In 2013, Carvajal et al. [12] apply these constraints to single-district instances from forest harvesting with roughly 350 to 1350 vertices. In 2017, Fischetti et al. [22] apply them to single-district instances of uniform Steiner tree and maximum-weight connected subgraph with thousands or tens of thousands of vertices. In 2017, Oehrlein and Haunert [39] apply them to multi-district instances with roughly 100 and 400 vertices. In the multi-district case, the variable $x_{ij}$ equals one when vertex $i$ is assigned to (the district rooted at) vertex $j$, and the $a, b$-separator constraints can be written in the form:

$$(a, b\text{-separator inequality, for multiple districts}) \qquad x_{ab} \leq \sum_{i \in S} x_{ib} \tag{7}$$

In 2022, Validi et al. [56] apply these constraints to multi-district instances with thousands of vertices and observe that a particular districting problem with a compactness objective is equally difficult with or without contiguity constraints.

### 2.2.3 Other models

In 1999, Drexl and Haase [20] proposed cut-based constraints for a sales districting problem. Before describing it, let us define a few notations. Given a vertex subset $S \subseteq V$, its open-neighborhood is given by $N(S) := (\cup_{i \in S} N(i)) \setminus S$. The closed-neighborhood is $N[S] := N(S) \cup S$. For a single vertex $i \in V$, we simply write $N[i]$ instead of the more cumbersome $N[\{i\}]$. Letting $x_{ij}$ be a binary variable that equals one when vertex $i$ is assigned to (the district rooted at) vertex $j$, the Drexl-Haase constraints can be written as:

$$\sum_{i \in N(S)} x_{ij} \geq 1 + \left( \sum_{i \in S} x_{ij} - |S| \right) \qquad \forall S \subseteq V \setminus N[j],\ \forall j \in V. \tag{8}$$

These constraints impose that, if all vertices of $S$ are assigned to vertex $j$, then at least one of $S$'s neighbors is as well. We can write a stronger form of these constraints that are invoked when *any* vertex $v \in S$ is assigned to $j$:

$$\sum_{i \in N(S)} x_{ij} \geq 1 + (x_{vj} - 1) \qquad \forall v \in S,\ \forall S \subseteq V \setminus N[j],\ \forall j \in V.$$

This is a special case of the $a, b$-separator inequalities, where $N(S)$ plays the role of the separator, and it may not be minimal. To our knowledge, this observation has not appeared in the literature, and researchers continue to use the weaker inequalities (8) rather than $a, b$-separators. We note that the tree-based, distance-based, and DAG-based constraints look almost like the $a, b$-separator inequalities (7), where the in-neighborhood acts as separator.

So far, we have discussed models that assign vertices to districts, with little to no focus on the edges. However, the literature contains many models that also use edge variables [28, 36, 37]. For example, Williams [59] proposes a linear-size model for spanning trees in planar graphs that is integral (see [41, 53] for a correction to this model), which can be adapted for districting [35, 60, 61]. However, the resulting models are not competitive with alternatives [61]. We point the reader to Rehfeldt et al. [43] for other models with nice theoretical properties that exploit connections to Steiner arborescence models.

# 3 Theoretical Analysis

Here, we analytically compare the tree-based, distance-based, and DAG-based models. We identify the sizes of these models, the number of solutions that they admit, the conditions under which they are integral, and the computational complexity of optimizing over them.

## 3.1 Basics

Before comparing these models, we should define them more formally. Each model can be defined in terms of a directed acyclic graph $H = (V, A)$, but for different choices of the edge set $A$. Below, we suggest how to obtain each such $A$.

Consider an input graph $G = (V, E)$ with a positive weight $w_e = w_{ij} = w_{ji}$ for each edge $e = \{i, j\} \in E$. Let $r$ be a root vertex and denote by $\text{dist}(r, j)$ the edge-weighted distance from $r$ to $j$ in $G$. For the tree-based constraints, let $A_{\text{tree}}$ be the edge set of a shortest path tree. For the distance-based constraints, we let

$$A_{\text{dist}} \coloneqq \bigcup_{i \in V \setminus \{r\}} \{(j, i) \mid j \in N(i) \text{ and } \text{dist}(r, j) = \text{dist}(r, i) - w_{ji}\}.$$

This picks up all edges that belong to any shortest path from $r$ to another vertex, whereas the tree-based constraints rely on a single shortest path's edges. Also, observe that $H_{\text{dist}} = (V, A_{\text{dist}})$ is a directed acyclic graph, and thus it admits a topological ordering [2, 16]. Construct $A_{\text{dag}}$ by orienting the edges of $E$ as per the topological ordering, i.e., edge $\{i, j\}$ is oriented as $(i, j)$ if $i$ comes before $j$ in the ordering—and $(j, i)$ if otherwise.

We make the following observation, which holds by construction.

**Lemma 1.** $A_{\text{tree}} \subseteq A_{\text{dist}} \subseteq A_{\text{dag}}$.

For a set of directed edges $A$, we define an associated polytope as follows,

$$P(A) \coloneqq \left\{ x \in [0,1]^n \;\middle|\; x_i \leq \sum_{j \in N^-(i)} x_j \quad \forall i \in V \setminus \{r\} \right\},$$

where $N^-(i)$ denotes the in-neighborhood of vertex $i \in V$ in graph $H = (V, A)$. The corresponding integer feasible set is given by

$$X(A) \coloneqq P(A) \cap \{0,1\}^n.$$

We have the following easy lemma.

**Lemma 2.** *If $A \subseteq A'$ then $X(A) \subseteq X(A')$.*

Denote by $X$ the "full" set of *all* contiguous solutions, i.e.,

$$X \coloneqq \{x^D \in \{0,1\}^n \mid G[D] \text{ is connected}\}. \tag{9}$$

Here, $x^D$ is the characteristic vector of $D$, which has $x_i^D = 1$ if and only if $i \in D$. By convention, we consider 0-vertex and 1-vertex graphs to be connected.

The following proposition formalizes the idea that the tree-based, distance-based, and DAG-based constraints permit only connected solutions, and the number of solutions grows as we move from left to right.

**Proposition 1.** $X(A_{\text{tree}}) \subseteq X(A_{\text{dist}}) \subseteq X(A_{\text{dag}}) \subseteq X$.

*Proof.* The first two inclusions hold by Lemmas 1 and 2. For the last inclusion, consider an integer point $x^* \in X(A_{\text{dag}})$. We show that $x^* \in X$, i.e., that $S \coloneqq \{i \in V \mid x_i^* = 1\}$ induces a connected subgraph. In the first case, where $S = \emptyset$, $G[S]$ is considered connected by convention. Otherwise, $S \neq \emptyset$, and it suffices to show that there exists a path in $G[S]$ connecting an arbitrary vertex $i$ from $S$ to root vertex $r$ (and that $r \in S$). This can be achieved with the following algorithm, which iteratively traces a path from $i$ back to $r$.

1. initialize path $P$ as the trivial sequence $[i]$
2. while $i \neq r$ do:
    - let $i$ be the first element of $P$
    - pick a vertex $j \in N^-(i)$ with $x_j^* = 1$ and do $P$.prepend($j$)

A suitable $j$ exists in each iteration of the while-loop by assumption that $x^* \in X(A_{\text{dag}})$ since $x_i^* \leq \sum_{j \in N^-(i)} x_j^*$. The algorithm terminates (and does so at $r$) because in each iteration $j$ comes before $i$ in the topological ordering. □

The LP relaxations $P(A)$ can be solved quickly given that there are only $n-1$ constraints (ignoring the 0-1 bounds), and these constraints have only $(n-1)+m$ nonzero coefficients. And, the graph $G$ is often planar, in which case the number of nonzeros is strictly less than $4(n-1)$ since planar graphs with at least three vertices have at most $3(n-2)$ edges. Further, $A_{\text{tree}}$ yields $2(n-1)$ nonzeros. So, for planar graphs, there is little difference between the tree-based, distance-based, and DAG-based models in terms of size—they are all within a factor 2 of each other—suggesting that the time needed to solve their LP relaxations will be substantially similar. These observations are summarized in Table 1.

Table 1: Size of models for graph $G = (V, E)$ with $n$ vertices and $m$ edges

| Contiguity | #Variables | #Constraints | #Nonzero Coefficients: for arbitrary $G$ | for planar $G$ |
|---|---|---|---|---|
| tree | $n$ | $n-1$ | $2n-2$ | $2n-2$ |
| dist | $n$ | $n-1$ | $\leq m+n-1$ | $\leq 4n-7$ |
| dag | $n$ | $n-1$ | $m+n-1$ | $\leq 4n-7$ |

## 3.2 Number of Solutions

Ideally, we would choose $A$ so that the model's integer feasible set $X(A)$ well-approximates the true feasible set $X$—or at least the portion of $X$ we are most interested in (e.g., reasonably compact districts). We examine this theoretically here and empirically in Section 4.

Specifically, we consider the number of solutions admitted by the tree-based, distance-based, and DAG-based constraints. First, we remark that there is a closed-form count for the tree-based constraints; however, for general DAGs this appears to be more difficult [42]. Next, we show that the number of solutions can grow substantially as we move from left to right in the string of inclusions $X(A_{\text{tree}}) \subseteq X(A_{\text{dist}}) \subseteq X(A_{\text{dag}}) \subseteq X$.

### 3.2.1 Exact count for tree-based constraints

We refer to an out-tree (a.k.a. an arborescence), which is a directed version of a tree in which all edges point away from the root vertex $r$. In an out-tree, there is precisely one directed path from $r$ to every other vertex.

Consider an out-tree $H = (V, A_{\text{tree}})$ rooted at $r$. Denote by $f(i)$ the number of connected solutions we can build using $i$ plus a subset of descendants. For a vertex $i$ with empty out-neighborhood $N^+(i) \coloneqq \{j \in V \mid (i,j) \in A\}$, this means $f(i) = 1$, since the only solution is $\{i\}$. For other vertices, we have the recursion

$$f(i) = \prod_{j \in N^+(i)} (f(j) + 1), \tag{10}$$

where the $+1$ term comes by *omitting* $j$ and its descendants. This recursion also applies to vertices with zero out-neighbors if we consider the empty product to equal one. The total number of tree-based solutions is then

$$|X(A_{\text{tree}})| = f(r) + 1, \tag{11}$$

where $f(r)$ can be computed recursively from its out-neighbors, and the $+1$ term represents the empty set (which we consider to be connected). So, for example, if $H = (V, A_{\text{tree}})$ is an out-star, then the number of solutions is

$$f(r) + 1 = \prod_{j \in N^+(i)} (f(j) + 1) + 1 = 2^{n-1} + 1.$$

In the other extreme, if $H = (V, A_{\text{tree}})$ is the path graph $(1, 2, 3, \ldots, n)$ with root $r = 1$, then the number of solutions is only

$$f(r) + 1 = (f(2) + 1) + 1 = \cdots = (f(n) + 1) + \underbrace{1 + 1 + \cdots + 1}_{n-1 \text{ times}} = n + 1.$$

This suggests that if we want $X(A_{\text{tree}})$ to have many solutions, then we should avoid long, snaking paths. Instead, $H = (V, A_{\text{tree}})$ should have many leaves, allowing for the internal vertices to have higher degree. This is intuitively what the distance-based constraints do, particularly under hop-based distances.

### 3.2.2 From polynomial to exponential size

By Proposition 1, we know that the number of solutions will generally *increase* as we move from the tree-based to the distance-based to the DAG-based constraints. The question we pose here is: How *many more* solutions? We show that the difference can be huge. The constructions that we use are all planar and use edge-weighted distances, where an edge's weight is the Euclidean distance between its endpoints.

**Theorem 1.** *For each inclusion $Y \subseteq Y'$ in the string $X(A_{\text{tree}}) \subseteq X(A_{\text{dist}}) \subseteq X(A_{\text{dag}}) \subseteq X$, there is a class of instances for which $|Y|$ has polynomial size but $|Y'|$ has exponential size.*

*Proof.* This follows by Lemmas 3, 4, and 5 below. □

First, we give a class of instances for which the tree-based constraints admit linearly many solutions, while the distance-based constraints admit exponentially many solutions.

**Lemma 3.** *There is a class of instances for which the tree-based constraints admit $|X(A_{\text{tree}})| = O(n)$ solutions, but the distance-based constraints admit $|X(A_{\text{dist}})| = \Omega(2^n)$.*

*Proof.* We construct a graph on $n \geq 1$ vertices. The vertices are located at $(x, y)$-coordinates of $(1, 0), (2, 0), \ldots, (n, 0)$. The root vertex is $(1, 0)$. The edges are chosen as in Figure 3. The weight of each edge is the Euclidean distance between its endpoints. We choose the shortest path tree as shown in the figure. The distance-based edges are also shown.

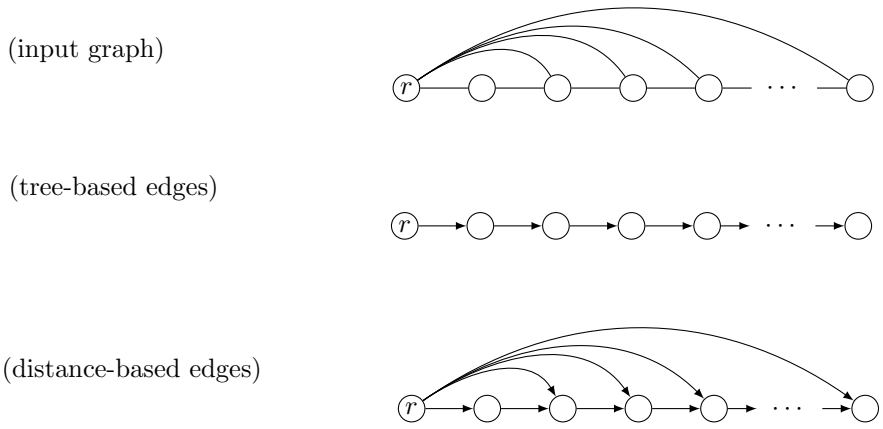


Figure 3: A class of instances with a linear number $O(n)$ of tree-based solutions and an exponential number $\Omega(2^n)$ of distance-based solutions.

Let us consider the number of solutions. For the tree-based constraints, we have the empty set and sets of the form $\{(1, 0), (2, 0), \ldots, (j, 0)\}$, yielding $1 + n = O(n)$ solutions. For the distance-based constraints, observe that the root vertex along with any subset of the other $n - 1$ vertices is feasible, yielding $2^{n-1} = \Omega(2^n)$ solutions. □

Next, we give a class of instances for which the distance-based constraints admit polynomially many solutions, while the DAG-based constraints admit exponentially many. This holds for either hop-based or edge-weighted distances—they are the same here. We remark that these instances structurally resemble parts of Iowa's county-level graph from Figure 1.

**Lemma 4.** *There is a class of instances for which the distance-based constraints admit* $|X(A_{\text{dist}})| = O(n^2)$ *solutions but the DAG-based constraints admit* $|X(A_{\text{dag}})| = \Omega(2^{n/2})$.

*Proof.* We construct a graph on $n = 2c+1$ vertices, where $c \geq 1$ is any positive integer. The vertices are located at $(x, y)$-coordinates of $(1, 0), (2, 0), \ldots, (c, 0)$ and $(1, 1), (2, 1), \ldots, (c, 1)$. There is also a special root vertex to the left, as shown in Figure 4, at unit distance from $(1, 0)$ and $(1, 1)$. The edges are also shown in figure. We use hop-based or Euclidean distances; they are the same here. The distance-based and DAG-based edges are shown in the figure.

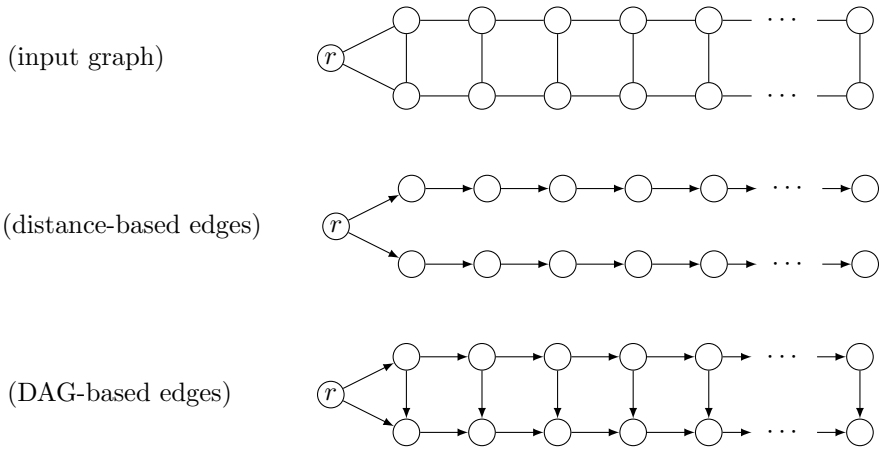


Figure 4: A class of graphs with a polynomial number $O(n^2)$ of distance-based solutions and an exponential number $\Omega(2^{n/2})$ of DAG-based solutions.

Let us consider the number of solutions. For the distance-based constraints, all nonempty solutions contain the root and some path along the top (perhaps none) plus some path along the bottom (perhaps none). Including the empty set, this gives $1 + (c + 1)(c + 1) = O(n^2)$ solutions. For the DAG-based constraints, observe that if the root and top row are chosen, then any subset of the other $c$ vertices is feasible to add, yielding $2^c = \Omega(2^{n/2})$ solutions. □

Last, we give a class of instances for which the DAG-based constraints admit linearly many solutions, while the full set $X$ has exponential size.

**Lemma 5.** *There is a class of instances for which the DAG-based constraints admit* $|X(A_{\text{dag}})| = O(n)$ *solutions, but the total number of contiguous solutions is* $|X| = \Omega(2^n)$.

*Proof.* We construct a graph on $n = c + 1$ vertices, where $c \geq 1$ is any positive integer. The vertices are located at $(x, y)$-coordinates of $(1, 0), (2, 0), \ldots, (c, 0)$ and $(c, 1)$. The root vertex is $(1, 0)$. The edges are chosen as in Figure 5. The weight of each edge is the Euclidean distance between its endpoints. Under this weighting, the DAG-based edges will point up and to the right, as shown.

We consider the number of solutions. For the DAG-based constraints, we have the empty set and sets of the form $\{(1, 0), (2, 0), \ldots, (j, 0)\}$ with or without the top vertex $(c, 1)$, giving

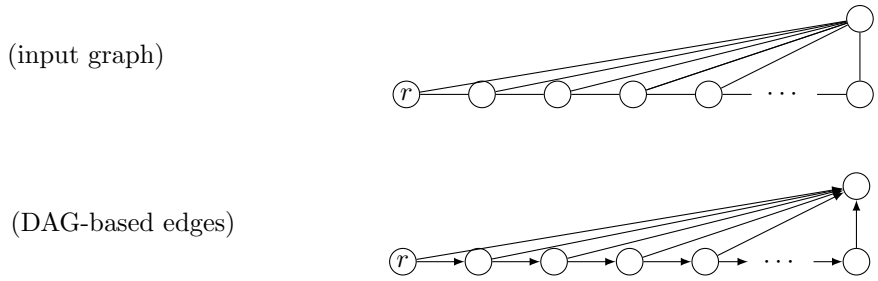


Figure 5: A class of instances with a linear number $O(n)$ of DAG-based solutions and an exponential number $\Omega(2^n)$ of contiguous solutions.

$1+2c = 1+2(n-1) = O(n)$ solutions. Now, if the root and top vertex are chosen, then any subset of the other $n-2$ vertices can be added, giving $2^{n-2} = \Omega(2^n)$ connected solutions. □

### 3.3 Integrality

We would like polytope $P(A)$ to well-approximate its integer hull $\text{conv}(X(A))$, so that branch-and-bound algorithms that use $P(A)$ as the LP relaxation will terminate quickly. The following theorem characterizes when $P(A)$ does so perfectly, i.e., $P(A) = \text{conv}(X(A))$

**Theorem 2.** *Let $H = (V, A)$ be a directed acyclic graph in which every vertex is reachable from root vertex $r$. The polytope $P(A)$ is integral if and only if $H$ is an out-tree.*

*Proof.* ( $\implies$ ) By the contrapositive. (See Figure 6.) Suppose that $H$ is not an out-tree, implying that there are at least two paths from $r$ to some other vertex $t$. Let $P_1$ and $P_2$ be distinct $r$-$t$ paths. We may assume that $t$ is the unique vertex that has two incoming edges in the subgraph induced by the edges of these paths (since otherwise we can locate an earlier vertex $t'$ in these paths and use the subpaths $P_1'$ and $P_2'$ from $r$ to $t'$ instead). We construct a fractional extreme point $x^*$ of $P(A)$ by setting $x^*_t = 1$, $x^*_i = 0.5$ for other $i \in V(P_1) \cup V(P_2)$, and $x^*_i = 0$ for the remaining vertices. It can be seen that this point

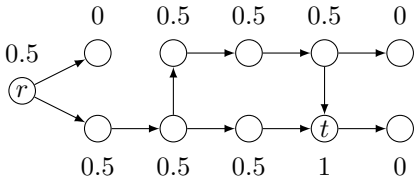


Figure 6: Illustration of extreme point $x^*$ for proof of Theorem 2.

belongs to $P(A)$, so it suffices to show extremality. This follows because $x^*$ is the unique point satisfying the following $n$ inequalities from $P(A)$ at equality.

$$
\begin{aligned}
& x_t \leq 1 && \\
& x_i \leq \sum_{j \in N^-(i)} x_j && \forall i \in V(P_1) \cup V(P_2) \setminus \{r\} \\
& x_i \geq 0 && \forall i \notin V(P_1) \cup V(P_2).
\end{aligned}
$$

($\Longleftarrow$) If $H$ is an out-tree, then each of its vertices $i \in V \setminus \{r\}$ has one in-neighbor, say, $\text{pred}(i)$, in which case $P(A)$ is the set of all $x \in [0,1]^n$ satisfying

$$x_i - x_{\text{pred}(i)} \leq 0 \qquad \forall i \in V \setminus \{r\}.$$

Since the associated constraint matrix is totally unimodular and the right-hand-sides are integral, integrality of $P(A)$ follows by well-known results [15, 32]. □

### 3.4 Complexity

Here we consider the computational complexity of optimizing over the tree-based, distance-based, and DAG-based constraints.

**Theorem 3.** *The integer optimization problem* $\max\{c^T x \mid x \in X(A)\}$ *is polynomial-time solvable for the tree-based constraints but NP-hard for the distance-based and DAG-based constraints (even when* $G - r$ *is planar).*

*Proof.* Polynomial-time solvability for $A = A_{\text{tree}}$ holds by Theorem 2 and the polynomiality of LPs. To show hardness for the other cases, we reduce from the vertex cover problem. This problem asks: Given a graph $G' = (V', E')$ and integer $k$, is there a vertex subset $S \subseteq V'$ with $|S| \leq k$ that touches every edge? This problem is NP-hard, even when $G'$ is planar [24]. Without loss of generality, we suppose $k < |V'|$ and $|E'| > 0$. We construct a graph $G = (V, E)$ by subdividing each edge of $G'$ and adding a new root vertex $r$ and connecting it to the original vertices of $G'$, i.e.,

$$\begin{aligned} V &\coloneqq V' \cup E' \cup \{r\} \\ E &\coloneqq \{\{r, v\} \mid v \in V'\} \cup \{\{v, e\} \mid e \in E',\ v \in e\}. \end{aligned}$$

By this construction, $G - r$ is planar. We orient the edges away from $r$ to obtain digraph $H = (V, A)$ where

$$A = \{(r, v) \mid v \in V'\} \cup \{(v, e) \mid e \in E',\ v \in e\}.$$

Note that $A$ is the set of distanced-based (or DAG-based) edges under hop-based distances. We set the objective coefficients for each $i \in V$ as follows:

$$c_i = \begin{cases} -1 & \text{if } i \in V' \\ |V'| + 1 & \text{if } i \in E' \\ 0 & \text{if } i = r. \end{cases}$$

Then, $G'$ admits a vertex cover of size at most $k$ if and only if the integer optimization problem has objective value at least $|E'|(|V'| + 1) - k$. □

Although the problem $\max\{c^T x \mid x \in X(A)\}$ is easy for the tree-based constraints, it is overly simplistic. Any real districting problem requires the populations of the districts to be similar. This is typically expressed via lower and upper bounds on the district population, i.e., $L \leq \sum_{i \in V} p_i x_i \leq U$, where $p_i$ is the population of vertex $i \in V$. In the presence of such population-balance constraints, the associated integer feasibility problem

$$\text{Is } \left\{ x \in X(A) \;\middle|\; L \leq \sum_{i \in V} p_i x_i \leq U \right\} \neq \emptyset?$$

is NP-complete, even for $A = A_{\text{tree}}$, as shown below. This suggests that, while these constraints might help in practice, they offer no relief in the worst case.

**Proposition 2.** *It is NP-complete to determine if the tree-based constraints admit a single population-balanced district, even for star graphs.*

*Proof.* We reduce from the knapsack problem. Given a list of integers $a_1, a_2, \ldots, a_n$ and target $b$, it asks to find an index subset $I \subset [n]$ for which $\sum_{i \in I} a_i = b$. We construct a star graph, where each leaf vertex $i \in [n]$ has population $p_i = a_i$ and the root vertex $r$ has zero population. Also, let $L = b$ and $U = b$. It is clear that the knapsack instance has a solution if and only if there is a binary point $x^*$ belonging to $X(A_{\text{tree}})$ with $L \leq \sum_{i \in V} p_i x_i^* \leq U$. $\square$

By essentially the same reduction, related optimization problems, like

$$\min_{x \in X(A)} \left\{ \sum_{i \in V} c_i x_i \;\middle|\; L \leq \sum_{i \in V} p_i x_i \right\} \text{ and } \max_{x \in X(A)} \left\{ \sum_{i \in V} c_i x_i \;\middle|\; \sum_{i \in V} p_i x_i \leq U \right\}$$

are NP-hard for the tree-based constraints.

The reduction above shows *weak* NP-hardness, requiring the use of "large" integers. However, what if the populations $p_i$ and thresholds $L$ and $U$ are all small integers like $p_i = 1$ and $L = U = 3$, what is the complexity then? We show that the associated *partitioning* problem is NP-hard, even if using the tree-based constraints for pre-defined roots of a planar graph. By contrast, the same problem is easy for $L = U = 2$ by matching.

**Theorem 4.** *It is NP-complete to determine if the tree-based constraints admit a contiguous partitioning into 3-vertex subsets, even for planar, bipartite graphs.*

*Proof.* To formalize things, introduce a binary variable $x_{ij}$ indicating whether vertex $i \in V$ is assigned to district $j \in [k]$. We suppose that district $j$ has a prescribed root vertex $r_j$ and denote by $A^j_{\text{tree}}$ the associated edge set for the tree-based constraints.

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad \forall i \in V \tag{12a}$$

$$L \leq \sum_{i \in V} p_i x_{ij} \leq U \qquad \forall j \in [k] \tag{12b}$$

$$[x_{ij}]_{i \in V} \in X(A^j_{\text{tree}}) \qquad \forall j \in [k] \tag{12c}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V,\ \forall j \in [k]. \tag{12d}$$

Here, constraints (12a) impose that each vertex is assigned to one district, constraints (12b) impose population balance, and constraints (12c) impose the tree-based constraints for each district. It is also permissible, but not required, to fix $r_j$ to be the root of district $j$ by setting $x_{r_j,j} = 1$.

We show it is NP-complete to determine if model (12) has a feasible solution, even when:

- the input graph $G = (V, E)$ is planar and bipartite;
- hop-based distances are used when computing the tree-based edges $A^j_{\text{tree}}$;
- each population $p_i$ equals one and $L = U = 3$.

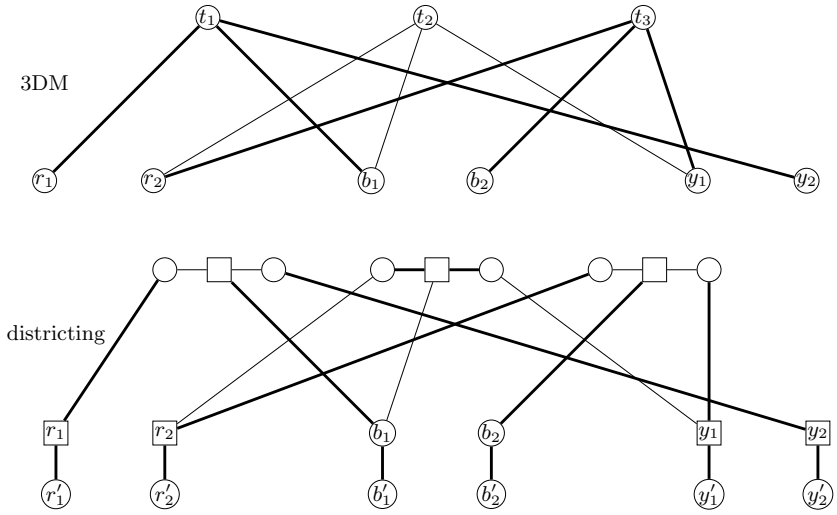


Figure 7: Reduction from 3DM to Districting (where square nodes denote roots). Solutions to each problem are shown with bold edges.

Dyer and Frieze [21] show that is NP-complete to determine whether the vertices of a graph $G = (V, E)$ can be partitioned into $|V|/3$ contiguous subsets, each with three vertices, see their Theorem 2.1. Their reduction is from planar 3-dimensional matching (3DM), which we recall below. Essentially, all that we must show is how to select the roots and then demonstrate that the tree-based constraints will permit a solution (if one exists).

In the 3DM problem, we are given three disjoint sets $R, B, Y$ of equal size and a collection of triples $T \subseteq R \times B \times Y$, and the task is to determine if there exists a subcollection $T' \subseteq T$ that hits each element of $R \cup B \cup T$ exactly once. In the *planar* 3DM problem, it is assumed that the graph $G' = (V', E')$ is planar, where $V' = R \cup B \cup Y \cup T$ and the edges $E'$ connect each vertex $t = (r, b, y) \in T$ to the associated elements $r \in R$, $b \in B$, and $y \in Y$. That is, $E' = \cup_{t=(r,b,y)\in T}\{\{t, r\}, \{t, b\}, \{t, y\}\}$.

From the planar 3DM instance $(R, B, Y, T)$, Dyer and Frieze construct a planar, bipartite graph $G = (V, E)$ as follows. For each $i \in R \cup B \cup Y$, create two vertices $i$ and $i'$ that are joined by an edge. For each $t = (r, b, y) \in T$, create three vertices $r_t$, $b_t$, and $y_t$, joined by edges to make a path $r_t - b_t - y_t$. Finally, for each $t = (r, b, y) \in T$, connect $r_t$ to $r$, $b_t$ to $b$, and $y_t$ to $y$. Figure 7 illustrates the reduction (but for a non-planar embedding).

The Dyer-Frieze reduction works as follows. If the planar 3DM instance has a solution $T' \subseteq T$, then we can obtain the connected partition of $G = (V, E)$ as follows. For each $t = (r, b, y) \in T$, if $t \in T'$ then use $\{r_t, r, r'\}$ and $\{b_t, b, b'\}$ and $\{y_t, y, y'\}$; otherwise use $\{r_t, b_t, y_t\}$. Contrariwise, if $G$ has a solution, then we can obtain a 3DM in reverse. Namely, for each $i \in R \cup B \cup Y$ the vertex $i'$ is a leaf in $G$ with stem $i$, so they must be in the same part. Whichever other vertex $i_t$ is in their part indicates a triple $t \in T$ for the 3DM.

Let $q = |R| = |B| = |Y|$ and $q' = |T|$ and see that $|V| = 6q + 3q'$. So, if we want to partition $V$ into connected subsets of 3 vertices each, there would be $k = 2q + q'$ districts. We select the following $k$ vertices as roots. Each $r \in R$ is a root. Each $y \in Y$ is a root. For each $t = (r, b, y) \in T$, $b_t$ is a root. These roots are drawn as squares in Figure 7. This works for our purposes, as follows.

If the 3DM instance has a solution, $G$ admits a connected partition into 3-vertex subsets. As discussed above, each subset $D$ takes one of the following forms: $\{r_t, r, r'\}$ or $\{b_t, b, b'\}$ or $\{y_t, y, y'\}$ or $\{r_t, b_t, y_t\}$. In the first and third cases, $r$ and $y$ can serve as the roots, respectively. In the second and fourth cases, $b_t$ can serve as the root. In all cases, $G[D]$ is a star with the root as hub, meaning that the tree-based constraints would allow $D$ (whether using hop-based distance or edge-weighted distances). So, model (12) is feasible.

Contrariwise, if model (12) admits a feasible solution, then it is a connected partition into 3-vertex subsets, implying that the planar 3DM instance has a solution. □

Theorem 4 reinforces the idea that the tree-based constraints offer no relief in the worst case. The next section explores the complexity "in practice" through computational experiments on real-life districting instances.

# 4 Computational Experiments

We perform computational experiments to shed light on the effectiveness and computational limits of the tree-based, distance-based, and DAG-based constraints. Specifically, we ask:

- Do enacted congressional and legislative districts across the USA actually satisfy any of the inexact contiguity constraints? How much does the answer depend on the edge weights used to calculate distances?
- How well do the inexact contiguity constraints approximate the "full" set of contiguous solutions (e.g., by feasible region size or by objective value)?
- How quickly can we optimize over the inexact contiguity constraints with present-day MIP solvers? How does the performance change as the instances grow in size? How do they compare to exact methods?

Computational experiments are run on a PC with Windows 11 enterprise, Intel Core i9-13900K CPU at 3.00 GHz (5.80 GHz turbo), and 64 GB RAM. The MIP solver is Gurobi v12.0.2. All code is written in Python, uses the NetworkX package, and is available at: `https://github.com/Ishaanjolly/A-guide-to-inexact-contiguity-constraints`

## 4.1 Do Enacted Districts Satisfy Inexact Contiguity?

In this section, we ask: What percentage of enacted congressional and legislative districts across the USA actually satisfy the inexact contiguity constraints?

The answer obviously depends on the choice of root and the edge-weighting scheme. For example, one "hack" to get a 100% success rate is to choose any root vertex $r$ from the district $D \subset V$, give district edges $e \in E \cap \binom{D}{2}$ a weight of one, and non-district edges $e \in E \setminus \binom{D}{2}$ a sufficiently large weight $M$. The district $D$ will satisfy the resulting tree-based, distance-based, and DAG-based constraints. However, this is not a realistic edge-weighting because it requires knowing the district $D$ a priori.

Instead, we consider four more reasonable edge-weighting schemes: Hop, Euclidean, HopM, and EuclideanM. In Hop, each edge $e \in E$ has weight $w_e = 1$. In Euclidean, each edge $e = \{i, j\} \in E$ has weight $w_e = \text{Eucl}(i, j)$ equal to the Euclidean distance between $i, j \in V$ on the map. In HopM, which is essentially the weighting scheme of Shahmizad and

Buchanan [47], we have

$$w_e = \begin{cases} 1 & \text{if } i,j \in V \text{ belong to same tract} \\ 1 + |V| & \text{if } i,j \in V \text{ belong to same county, but different tracts} \\ 1 + |V|^2 & \text{if } i,j \in V \text{ belong to different counties,} \end{cases}$$

where $|V|$ plays the role of a big-$M$. Last, we propose a Euclidean generalization called EuclideanM, where $M$ is chosen sufficiently large, setting:

$$w_e = \begin{cases} \text{Eucl}(i,j) & \text{if } i,j \in V \text{ belong to same tract} \\ \text{Eucl}(i,j) + M & \text{if } i,j \in V \text{ belong to same county, but different tracts} \\ \text{Eucl}(i,j) + M^2 & \text{if } i,j \in V \text{ belong to different counties.} \end{cases}$$

We still require a root vertex. Lacking a better choice, we compute the population-weighted mean of each district $D$ in Euclidean space, and select the nearest vertex to be the root. From this vertex $r \in V$, we compute edge-weighted distances to set the tree-based, distance-based, and DAG-based constraints. By using edge-weighted distances in $G$, every vertex will be reachable from $r$ in the resulting digraph $H = (V, A)$, assuming $G$ is connected. This would generally not be true if we simply oriented each edge $\{i, j\}$ towards whichever vertex $i, j \in V$ was farther from $r$ on the map (e.g., for non-convex shapes).

We collect the 2022 congressional and legislative districting plans from the US Census Bureau [51, 52], stored as block assignment files (BAFs), and check each district to see whether it satisfies the inexact contiguity constraints under the different edge-weighting schemes. Results are provided in Table 2. In total, there are 7265 enacted districts, of which 168 are actually disconnected in the associated block-level graph $G$. So, when calculating percentages we use $7265 - 168 = 7097$ in the denominator.

Table 2: How many enacted districts satisfy inexact contiguity?

| | Edge-Weighting Scheme | | | |
|---|---|---|---|---|
| Contiguity | Hop | Euclidean | HopM | EuclideanM |
| tree | 161 ( 2.27%) | 406 ( 5.72%) | 702 ( 9.89%) | 862 (12.15%) |
| dist | 430 ( 6.06%) | 406 ( 5.72%) | 973 (13.71%) | 862 (12.15%) |
| dag | 752 (10.60%) | 1593 (22.45%) | 1794 (25.28%) | 2040 (28.74%) |

As expected, the percentages increase as we move from tree-based to distance-based to DAG-based constraints. For example, for the hop-based weighting scheme, the percentages are 2.27%, 6.06%, and 10.60%, respectively. However, for the Euclidean and EuclideanM schemes, there is no difference between the tree-based and distance-based counts, presumably because the shortest paths will tend to be unique in this regime, in which case the distance-based edge set $A_{\text{dist}}$ will be an out-tree, the same as $A_{\text{tree}}$. Thus, it may come as no surprise that the distance-based constraints actually permit *fewer* districts under the Euclidean weightings than under hop-based weightings.

We notice that the big-$M$ schemes outperform their no-$M$ counterparts, often by 100% or more. For example, when moving from Hop to HopM, the tree-based count jumps from 161 to 702, and the DAG-based count jumps from 752 to 1794. The tree-based and DAG-based constraints also benefit from the Euclidean weighting schemes, although slightly less so than from the big-$M$ schemes, e.g., 161 to 406 for tree-based and 752 to 1593 for DAG-based.

However, both perform best under the combined EuclideanM weighting scheme, with the DAG-based count reaching 2040, or 28.74% of the enacted districts. This appears to be the best setting among the bunch if seeking to recover the type of district that is drawn in practice. However, we must not forget that this setting still "overlooks" more than 70% of all enacted districts, suggesting that exact contiguity constraints may still be needed.

## 4.2 Enumeration for Grid Graphs

In this section, we enumerate all plans that satisfy the inexact and exact contiguity constraints for small grid graphs, 4×4 and 6×6. We use both square grid graphs and triangular grid graphs, as depicted in Figure 8. Real-life graphs, like Iowa's county-level graph from Figure 1, are often a mix of these two extremes [3].

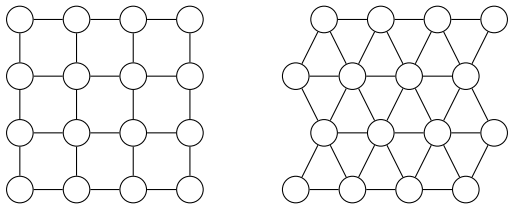

Figure 8: 4×4 square grid and 4×4 triangular grid.

For simplicity, we draw $k = 4$ districts, fixing the four corner vertices as roots, and use hop-based distances to define the inexact contiguity constraints. We require the districts to have equal numbers of vertices. The resulting model is similar to (12), but with unit populations. With such a simple setup, we can enumerate *all* feasible plans in a reasonable amount of time using Gurobi's PoolSearchMode. Table 3 gives results, reporting the number of feasible solutions and the total runtime.

Table 3: Enumeration results for grid graphs

| | | Square Grids | | Triangular Grids | |
|---|---|---|---|---|---|
| Size | Contiguity | # Soln | Time | # Soln | Time |
| 4×4 | tree | 7 | 0.026 | 20 | 0.024 |
| | dist | 33 | 0.025 | 61 | 0.030 |
| | dag | 33 | 0.022 | 65 | 0.029 |
| | exact | 33 | 0.068 | 71 | 0.083 |
| 6×6 | tree | 218 | 0.038 | 1,077 | 0.074 |
| | dist | 45,107 | 11.050 | 55,654 | 15.755 |
| | dag | 45,017 | 11.472 | 152,746 | 113.570 |
| | exact | 82,419 | 185.745 | 452,196 | 3511.828 |

As expected from Proposition 1, the number of solutions generally grows as we move from tree-based to distance-based to DAG-based and finally to the exact (separator-based) constraints. However, on the 4x4 square grid, the distance-based constraints recover all solutions, leaving no room for the DAG-based or exact constraints to outshine them.

On the 6x6 square grid, the tree-based constraints permit only a small fraction of the possible solutions, just 218/82419 or 2.6%, while the distance-based and DAG-based constraints

recover 54.7% of the solutions. On larger square grids, the distance-based and DAG-based constraints would continue to be identical, given that these graphs are bipartite, in which case the distance-based constraints already use every edge under the hop-based scheme.

The situation changes for the triangular grid graphs, most notably for the 6x6 case. Like before, the tree-based constraints recover only a small fraction of the connected solutions, 1077/452196 or 0.2%. There is a sizable increase as we transition to the distance-based constraints, jumping to 12.3%, and the DAG-based constraints further triple this to 33.8%.

As a last remark, if the roots are free-floating, not fixed in the corners, then the number of solutions grows to 117 for the 4x4 square grid, and 442,791 for the 6x6 square grid [46]. Generally, the number of ways to partition $k \times k$ square grids into $k$ equal-size districts grows exponentially in $n = k^2$, roughly between $1.41^n$ and $3.21^n$, as proven by [19]. We are not aware of similar analyses for triangular grids.

## 4.3 Enumeration for Iowa

In the previous section, we considered the stylized, synthetic case of grid graphs with unit populations. In this section, we perform a similar enumeration analysis, but using the real state of Iowa, which keeps all counties whole in its $k = 4$ congressional districts. Similar to before, we select its four corner counties (Allamakee, Fremont, Lee, Lyon) as roots. When defining the inexact contiguity constraints, we apply the Hop and Euclidean schemes from Section 4.1. (There is no reason to use HopM and EuclideanM when working with county-level graphs.) We apply deviations of $\pm d$ people for $d \in \{10, 20, 30, 40, 50\}$, similar to the enacted map's $\pm 53$-person deviation. Table 4 gives results under a 3600s time limit.

Table 4: Enumeration results for Iowa (* indicates timeout)

| | | Number of feasible solutions, by deviation | | | | |
|---|---|---|---|---|---|---|
| Scheme | Contiguity | ±10 | ±20 | ±30 | ±40 | ±50 |
| Hop | tree | 0 | 0 | 0 | 0 | 0 |
| | dist | 4 | 61 | 210 | 487 | 954 |
| | dag | 199* | 678* | 4,039* | 6,986* | 17,273* |
| Euclidean | tree | 0 | 0 | 0 | 0 | 0 |
| | dist | 0 | 0 | 0 | 0 | 0 |
| | dag | 138* | 1,110* | 2,282* | 3,290* | 12,353* |

For Iowa, we see that the tree-based constraints are too restrictive, to the point of being worthless. Not even a single districting plan satisfies them, at least at population deviations that are typically used in practice. This holds true for both edge-weighting schemes.

Transitioning to the distance-based constraints slightly improves the situation. The hop-based scheme permits 4 plans at a ±10-person deviation and 954 plans at a ±50-person deviation. However, the Euclidean scheme allows no plans whatsoever, presumably because the shortest paths are unique, causing the distance-based constraints to collapse to the tree-based constraints, just as we saw in Section 4.1.

The DAG-based constraints really shine at these tight deviations, already allowing hundreds or thousands of solutions at population deviations of ±10 and ±30 people, respectively, regardless of the edge-weighting scheme. The number of solutions exceeds 10,000 at a ±50-person deviation, significantly more than allowed by the distance-based constraints. This shows that the edges ignored by the distance-based constraints (i.e., those edges $\{i, j\} \in E$

with $\text{dist}(r,i) = \text{dist}(r,j)$ in the hop-based scheme) are being put to good use by the DAG-based constraints. Overall, the results suggest that the DAG-based constraints are preferable (as compared to the tree-based and distance-based constraints) when the problem is highly-constrained, as is true for county-level districting under tight population tolerances.

## 4.4 Optimization for Iowa

In the previous section, we considered the highly-constrained setting of county-level districting, where we had some chance of enumerating all feasible solutions. In this section, we turn to optimization, where we seek a single best solution. This allows us to consider larger, more granular instances, to understand the relative advantages and disadvantages of the different contiguity constraints in such settings.

For our experiments, we impose a deviation of $\pm 50$ people, similar to Iowa's enacted map, and force the four districts to be rooted at the four corners of the state (i.e., northeast, northwest, southeast, and southwest). We consider three different levels: counties ($n = 99$), voting precincts ($n = 2{,}536$), and census blocks ($n = 175{,}199$). We apply the four different edge-weighting schemes (Hop, HopM, Euclidean, EuclideanM) to the tree-based, distance-based, and DAG-based constraints. For benchmarking purposes, we also compare these inexact contiguity models against the exact (separator) constraints and against a setting in which contiguity constraints are omitted entirely. This helps us ascertain whether contiguity constraints truly make the problem more difficult than otherwise and whether inexact contiguity constraints truly offer a computational advantage over existing exact constraints.

We consider two popular objectives from the literature. The first objective is the classical moment-of-inertia objective of Hess et al. [31], which resembles popular facility location objectives and takes the form

$$\text{(Moment-of-Inertia Objective)} \qquad \min \sum_{i \in V} \sum_{j \in V} d_{ij}^2 p_i x_{ij},$$

where $d_{ij}$ is the Euclidean distance between (the centers of) geographic units $i$ and $j$, $p_i$ is the population of $i$, and $x_{ij}$ is a binary variable indicating whether $i$ is assigned to (the district centered at) $j$. The second objective is the number of cut edges between the districts [55], which takes the form

$$\text{(Cut Edges Objective)} \qquad \min \sum_{e \in E} y_e,$$

where $y_e$ is a binary variable indicating whether the endpoints of edge $e \in E$ are assigned to different districts. This objective requires additional coupling constraints to link the district assignment variables $x_{ij}$ to the cut edge variables $y_e$ with $e = \{u, v\}$, taking the form

$$x_{uj} - x_{vj} \leq y_e \quad \text{and} \quad x_{vj} - x_{uj} \leq y_e.$$

The other constraints of the models impose population balance and partitioning, as in (12).

Table 5 gives county-level results, reporting the optimal objective value, the runtime, and the number of branch-and-bound nodes explored. If an instance was not solved to proven optimality, we report the best lower and upper bounds [LB,UB] at termination, where an objective value of $+\infty$ corresponds to infeasibility. Recall that, at the county level, it is

difficult to satisfy the imposed population balance constraints, as they only allow a ±50-person deviation. Compare this to the populations of Iowa's counties, which range from 3,704 (Adams County) to 492,401 (Polk County). This provides some insight into why the tree-based constraints never find a feasible plan.

Likewise, the distance-based constraints allow no plans when using the Euclidean edge-weighting scheme. However, under the hop-based scheme, they allow 954 plans, as we saw in Table 4. Across these 954 plans, the best moment-of-inertia was roughly 8.26 million, which is 3.73% larger than the true optimal objective value 7.96 million. Meanwhile, as we transition to the DAG-based constraints, this drops to only 0.51% and 0.11% for the hop-based and Euclidean schemes, respectively. This shows that the DAG-based constraints allow not only *more* solutions than other inexact contiguity constraints, but also *better* solutions. The situation is similar for the cut edges objective, where under the hop-based scheme the distance-based constraints find a plan with 51 cut edges, while the DAG-based constraints noticeably improves this to 44.

Interestingly, if contiguity constraints are omitted entirely, then the MIP solver still struggles to prove optimality within the one-hour time limit, terminating with bounds of $[36, 41]$. Contrast this with the moment-of-inertia objective, which the exact model solved in half a second. This suggests that the cut edges objective function is already difficult by itself; it is not merely the population balance or contiguity constraints causing trouble.

Table 5: Optimization Results for Iowa at County-Level, where $|V| = 99$

| | | Moment of Inertia | | | Cut Edges | | |
|---|---|---|---|---|---|---|---|
| Scheme | Contiguity | Objective | Time(s) | #BB | Objective | Time(s) | #BB |
| Hop | tree | $+\infty$ | 2.16 | 11,227 | $+\infty$ | 4.02 | 10,704 |
| | dist | 8,261,283.85 | 137.89 | 758,217 | 51 | 730.98 | 1,388,643 |
| | dag | 8,004,829.45 | 28.55 | 97,724 | 44 | 2,204.67 | 1,390,955 |
| Euclidean | tree | $+\infty$ | 1.91 | 5,542 | $+\infty$ | 1.44 | 3,926 |
| | dist | $+\infty$ | 1.48 | 5,542 | $+\infty$ | 1.42 | 3,926 |
| | dag | 7,972,386.64 | 39.77 | 255,396 | [40,51] | TL | 1,943,598 |
| N/A | none | 7,800,525.95 | 0.46 | 8,322 | [36,41] | TL | 2,466,633 |
| | exact | 7,963,865.85 | 1,134.91 | 6,212,588 | [31,$+\infty$] | TL | 1,201,181 |

Table 6 gives precinct-level results. Here we have substantially more granularity than before, with 2,536 precincts instead of 99 counties. Despite the much larger graph size, these instances are solved in just a matter of seconds when using the moment-of-inertia objective. In some sense, all inexact contiguity constraints shine, yielding objective values that lie within 0.33% of the true optimum, with the worst-case being the tree-based constraints under the hop-based scheme (9,297,763.94 tree-based vs. 9,266,931.76 exact). The DAG-based constraints produce solutions that are *extremely* close to optimal; namely, the Euclidean scheme arrives at 9,266,932.49, or 0.0000079% larger than the true optimum. For some comparison, this is significantly smaller than the default optimality gap 0.01% used by Gurobi. Although the inexact contiguity constraints do work well here, one could argue that the exact constraints would be preferable, since they solve in only five seconds. We also observe that there is a difference in objective value between the exact model and the model without contiguity constraints, demonstrating that even with the substantial granularity offered by the 2,536 precincts, we may still require explicit contiguity constraints; the compactness-seeking objective alone is insufficient to guarantee this.

We see a marked increase in difficulty as we transition over to the cut edges objective. For

example, not even the "easy" tree-based constraints terminate within 10 minutes. Moreover, the best objective value that is found comes from the tree-based constraints. Not even the model *that lacks contiguity constraints* identifies a solution better than this within the time limit, only reaching 129 cut edges. Moreover, the DAG-based constraints and exact contiguity constraints never even find a feasible solution.

Table 6: Optimization Results for Iowa at Precinct-Level, where $|V| = 2{,}536$

| | | Moment of Inertia | | | Cut Edges | | |
|---|---|---|---|---|---|---|---|
| Scheme | Contiguity | Objective | Time(s) | #BB | Objective | Time(s) | #BB |
| Hop | tree | 9,297,763.94 | 2.56 | 234 | 113 | 1,507.51 | 9,540 |
| | dist | 9,272,390.74 | 1.98 | 3,978 | [94,113] | TL | 4,255 |
| | dag | 9,267,450.20 | 2.47 | 12,368 | [20,+∞] | TL | 270 |
| Euclidean | tree | 9,267,021.49 | 2.45 | 675 | 114 | 833.56 | 1,372 |
| | dist | 9,267,021.49 | 2.47 | 675 | 114 | 838.54 | 1,372 |
| | dag | 9,266,932.49 | 1.15 | 5,387 | [17,+∞] | TL | 222 |
| HopM | tree | 9,273,516.58 | 2.61 | 3,946 | 113 | 644.44 | 2,072 |
| | dist | 9,271,234.16 | 1.26 | 239 | [92,113] | TL | 1,313 |
| | dag | 9,267,711.09 | 1.39 | 7,479 | [19,+∞] | TL | 268 |
| EuclideanM | tree | 9,272,880.44 | 2.42 | 226 | 113 | 830.40 | 520 |
| | dist | 9,272,880.44 | 2.31 | 226 | 113 | 833.32 | 520 |
| | dag | 9,267,711.09 | 1.10 | 5,391 | [24,+∞] | TL | 272 |
| N/A | none | 9,266,904.94 | 0.21 | 955 | [14,129] | TL | 1,820 |
| | exact | 9,266,931.76 | 5.54 | 4,489 | [22,+∞] | TL | 549 |

Table 7 gives block-level results. These instances are incredibly large and granular, having 175,199 census blocks. Recall that census blocks often correspond to city blocks in urban areas, but can be geographically larger in rural areas. Many blocks have single-digit or zero-person populations, making it trivial to satisfy the ±50-person deviation constraints. Nevertheless, none of the models found a feasible solution when the cut edges objective was used to drive the search. Moreover, in all cases, the best lower bound found within the time limit was the trivial bound of zero. This shows that the linear programming relaxation is extremely weak, which explains the poor performance of the cut edge models. It appears to have little or nothing to do with the contiguity constraints themselves.

Perhaps surprisingly, the moment-of-inertia models continue to solve to optimality in just a matter of minutes. Only the exact model failed to solve, terminating before finding a feasible solution or even solving the root LP relaxation. Another surprising phenomenon is that the solve times across the inexact contiguity models are typically in reverse order of their size. That is, the largest model (DAG-based) usually solves more quickly than the distance-based model, which usually solves more quickly than the tree-based model. For example, under the hop-based scheme, the solve times for the tree-based, distance-based, and DAG-based models are 517.72, 302.58, and 238.17 seconds, respectively. Many of them solve at the root branch-and-bound node. Intuitively, this is because the difference between the root LP relaxation and the optimal IP objective is miniscule. For example, the root LP bound for the model that lacks contiguity constraints is 9,830,589.01, which is only a hair below the optimal IP objective of 9,830,589.07. The root LP bounds for the contiguity-constrained models cannot be less, so if they admit integer solutions with similar objective values, then the branch-and-bound nodes can be pruned almost immediately, perhaps after a bit of reduced-cost fixing. For example, under the Euclidean scheme, the DAG-based

constraints produce an integer solution with objective value 9,830,589.55, which is just a hair above its root LP bound of 9,830,589.37. Overall, the "price of contiguity" is very small, with little difference between the objective values with and without contiguity constraints, as also observed by Grover et al. [29] for a similar problem.

Table 7: Optimization Results for Iowa at Block-Level, where $|V| = 175{,}199$

| | | Moment of Inertia | | | Cut Edges | | |
|---|---|---|---|---|---|---|---|
| Scheme | Contiguity | Objective | Time(s) | #BB | Objective | Time(s) | #BB |
| Hop | tree | 9,832,889.61 | 517.72 | 1 | [0,+∞] | TL | 1 |
| | dist | 9,832,572.52 | 302.58 | 2,578 | [0,+∞] | TL | 1 |
| | dag | 9,830,793.94 | 238.17 | 1 | [0,+∞] | TL | 1 |
| Euclidean | tree | 9,830,591.07 | 115.95 | 60 | [0,+∞] | TL | 1 |
| | dist | 9,830,591.07 | 98.19 | 60 | [0,+∞] | TL | 1 |
| | dag | 9,830,589.55 | 122.65 | 1 | [0,+∞] | TL | 1 |
| HopM | tree | 9,834,331.91 | 639.08 | 894 | [0,+∞] | TL | 1 |
| | dist | 9,831,324.35 | 186.89 | 1 | [0,+∞] | TL | 1 |
| | dag | 9,830,663.09 | 145.86 | 1 | [0,+∞] | TL | 1 |
| EuclideanM | tree | 9,831,084.10 | 292.51 | 1 | [0,+∞] | TL | 1 |
| | dist | 9,831,084.10 | 294.68 | 1 | [0,+∞] | TL | 1 |
| | dag | 9,830,633.93 | 135.77 | 1 | [0,+∞] | TL | 1 |
| N/A | none | 9,830,589.07 | 13.18 | 1 | [0,194027] | TL | 1 |
| | exact | [−∞,+∞] | TL | 0 | [0,+∞] | TL | 1 |

# 5 Conclusion

This paper is motivated by the prevalence of inexact contiguity constraints in the integer programming and operations research literature. Despite their wide use, the tree-based, distance-based, and DAG-based constraints had not previously been compared, computationally or analytically, and their key structural properties (e.g., solution counts, integrality conditions, complexity) were left unstated. This paper fills that gap.

By applying these contiguity constraints to real-life districting instances, we now better understand their relative strengths and weaknesses in practice, which provide valuable insight to future researchers and practitioners. Our biggest takeaways are as follows.

1. On highly-constrained instances (e.g., county-level districting with small population deviations), the tree-based constraints are practically useless, and the distance-based constraints do not help much either. In such situations, DAG-based constraints and/or exact constraints are preferable given their ability to capture more solutions.

2. On very large instances (e.g., block-level graphs with hundreds of thousands of vertices), all inexact contiguity constraints are reasonable choices, particularly when the LP relaxation is expected to be strong (e.g., for transportation-like objectives). They all achieve essentially the same objective values, but, surprisingly, the largest model (DAG-based) generally solves the quickest and terminates at the root node.

3. Even for the largest, most granular instances, compactness objectives often do not automatically produce contiguous solutions as a byproduct. Explicit contiguity constraints are still needed.

4. On moderately-sized instances (e.g., precinct-level graphs with a few thousand vertices) with challenging objective functions (e.g., cut edges), the tree-based and distance-based constraints may be preferable. In such circumstances, the DAG-based constraints permit too many solutions to sift through when the LP relaxation is weak, making it unable to quickly prune the nodes of the branch-and-bound tree. However, this failure should not be attributed to the difficulty of imposing contiguity; the same models without contiguity constraints flounder in such situations.

5. The edge-weighting scheme can have a huge practical effect. Especially for distance-based constraints, the hop-based schemes generally lead to the most feasible solutions, whereas Euclidean schemes tend to yield unique shortest paths, causing the distance-based constraints to collapse to the tree-based constraints, thus providing no advantage.

6. Relatively few enacted congressional and state legislative districts across the USA satisfy the inexact contiguity constraints, regardless of the edge-weighting scheme. This number is as small as 2.27% for the tree-based constraints under the Hop scheme, and as large as 28.74% for the DAG-based constraints under the EuclideanM scheme. This suggests that, if one wants to draw the kinds of districts that are enacted in practice, it is unrealistic to directly apply inexact contiguity constraints.

Future work may investigate extensions of the DAG-based model that permit a select number of "back edges" that point upstream, against the topological ordering. Is there a computationally effective way to model this, that is substantially faster than exact methods? Are there substantial improvements to be had in terms of the number of solutions or objective values in highly-constrained settings?

Another general technique that may warrant further investigation is the use of aggregation or coarsening. For example, Swamy et al. [50] repeatedly find graph matchings, merging the endpoints of each matching edge into a single super node, until the graph has been reduced to a size manageable for exact methods. Would it be advantageous to stop the coarsening at a larger scale and and instead apply inexact contiguity constraints? Or, can we intelligently aggregate smaller geographic units into larger ones (e.g., census blocks into a tract, or census tracts into a county) before applying exact or inexact models? Is there an advantage to using inexact contiguity constraints in a local search procedure, akin to that of Belotti et al. [7], in which the cores of districts are coarsened into super nodes and the peripheral vertices that lie within $h$ hops of a district boundary are allowed to relocate?

**Acknowledgments.** We thank Daryl DeFord for sharing the districting instances. This material is based on work supported by the National Science Foundation (Grant No. 1942065) and the Air Force Office of Scientific Research (Grant No. FA9550-25-1-0277).

# References

[1] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization: Festschrift for Martin Grötschel*, pages 449–481. Springer, 2013.

[2] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

[3] Sara Anderson, Sarah Cannon, Brooke Feinberg, and Anne Friedman. Census dual graphs: Properties and random graph models, 2026.

[4] Christina Backes, Alexander Rurainski, Gunnar W Klau, Oliver Müller, Daniel Stöckel, Andreas Gerasch, Jan Küntzer, Daniela Maisel, Nicole Ludwig, Matthias Hein, et al. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, 40(6):e43–e43, 2012.

[5] Marc Bailly-Bechet, Christian Borgs, Alfredo Braunstein, J Chayes, A Dagkessamanskaia, J-M François, and Riccardo Zecchina. Finding undetected protein associations in cell signaling by belief propagation. *Proceedings of the National Academy of Sciences*, 108(2):882–887, 2011.

[6] Amariah Becker and Justin Solomon. Redistricting algorithms. In Moon Duchin and Olivia Walch, editors, *Political Geometry*, pages 303–340. Birkhauser, 2022.

[7] Pietro Belotti, Austin Buchanan, and Soraya Ezazipour. Political districting to optimize the Polsby-Popper compactness score with application to voting rights. *Operations Research*, 73(5):2330–2350, 2025.

[8] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121, 2012.

[9] Austin Buchanan. Political districting. In Panos M. Pardalos and Oleg A. Prokopyev, editors, *Encyclopedia of Optimization*. Springer International Publishing, Cham, 2023.

[10] Austin Buchanan, Je Sang Sung, Sergiy Butenko, and Eduardo L Pasiliao. An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing*, 27(1):178–188, 2015.

[11] Alexander Butsch, Jörg Kalcsics, and Gilbert Laporte. Districting for arc routing. *INFORMS Journal on Computing*, 26(4):809–824, 2014.

[12] Rodolfo Carvajal, Miguel Constantino, Marcos Goycoolea, Juan Pablo Vielma, and Andrés Weintraub. Imposing connectivity constraints in forest planning models. *Operations Research*, 61(4):824–836, 2013.

[13] Chao-Yeh Chen and Kristen Grauman. Efficient activity detection with max-subgraph search. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1274–1281. IEEE, 2012.

[14] Jeanne Clelland, Haley Colgate, Daryl DeFord, Beth Malmskog, and Flavia Sancier-Barbosa. Colorado in context: Congressional redistricting and competing fairness criteria in Colorado. *J Computational Social Science*, 5(1):189–226, 2022.

[15] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer, Cham, 2014.

[16] Thomas H Cormen, Charles E Leiserson, RL Rivest, and C Stein. *Introduction to Algorithms*. MIT Press, Cambridge, 2001.

[17] Thomas J Cova and Richard L Church. Contiguity constraints for single-region site search problems. *Geographical Analysis*, 32(4):306–329, 2000.

[18] Marcus T Dittrich, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

[19] Christopher Donnay and Matthew Kahle. Asymptotics of redistricting the $n \times n$ grid. *The American Mathematical Monthly*, 132(9):856–866, 2025.

[20] Andreas Drexl and Knut Haase. Fast approximation methods for sales force deployment. *Management Science*, 45(10):1307–1323, 1999.

[21] Martin E Dyer and Alan M Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139–153, 1985.

[22] Matteo Fischetti, Markus Leitner, Ivana Ljubić, Martin Luipersbeck, Michele Monaci, Max Resch, Domenico Salvagnin, and Markus Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Math Prog Comput*, 9(2):203–229, 2017.

[23] Tetsuya Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43(4):212–223, 2004.

[24] Michael R Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[25] Robert S Garfinkel and George L Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B–495, 1970.

[26] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

[27] Sebastian Goderbauer and Jeff Winandy. Political districting problem: Literature review and discussion with regard to federal elections in Germany, 2018.

[28] Michel X Goemans and Young-Soo Myung. A catalog of Steiner tree formulations. *Networks*, 23(1):19–28, 1993.

[29] Nidhima Grover, Xiaoyan Si, and Alejandro Toriello. The fulfillment regionalization problem. Technical report, Optimization Online, `https://optimization-online.org/2025/12/the-fulfillment-regionalization-problem/`, 2025.

[30] Knut Haase and Sven Müller. Upper and lower bounds for the sales force deployment problem with explicit contiguity constraints. *European Journal of Operational Research*, 237(2):677–689, 2014.

[31] SW Hess, JB Weaver, HJ Siegfeldt, JN Whelan, and PA Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965.

[32] Alan J Hoffman and Joseph B Kruskal. Integral boundary points of convex polyhedra. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 49–76. Springer, 2009.

[33] Nahid Jafari and John Hearne. A new method to solve the fully connected reserve network design problem. *European Journal of Operational Research*, 231(1):202–209, 2013.

[34] Zeyad Kassem and Adolfo Escobedo. The edge-based contiguous $p$-median problem with connections to logistics districting, 2026.

[35] Myung Kim and Ningchuan Xiao. Contiguity-based optimization models for political redistricting problems. *International Journal of Applied Geospatial Research (IJAGR)*, 8(4):1–18, 2017.

[36] Ivana Ljubić. Solving Steiner trees: Recent advances, challenges, and perspectives. *Networks*, 77(2):177–204, 2021.

[37] Thomas L. Magnanti and Laurence A. Wolsey. Chapter 9 optimal trees. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503–615. Elsevier, 1995.

[38] Anuj Mehrotra, Ellis L Johnson, and George L Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1114, 1998.

[39] Johannes Oehrlein and Jan-Henrik Haunert. A cutting-plane method for contiguity-constrained spatial aggregation. *Journal of Spatial Information Science*, 2017(15):89–120, 2017.

[40] Hayri Önal, Yicheng Wang, Sahan TM Dissanayake, and James D Westervelt. Optimal design of compact and functionally contiguous conservation management areas. *European Journal of Operational Research*, 251(3):957–968, 2016.

[41] Kanstantsin Pashkovich. *Extended formulations for combinatorial polytopes*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2012.

[42] Yisu Peng, Yuxiang Jiang, and Predrag Radivojac. Enumerating consistent sub-graphs of directed acyclic graphs: an insight into biomedical ontologies. *Bioinformatics*, 34(13):i313–i322, 2018.

[43] Daniel Rehfeldt, Henriette Franz, and Thorsten Koch. Optimal connected subgraphs: Integer programming formulations and polyhedra. *Networks*, 80(3):314–332, 2022.

[44] Federica Ricca, Andrea Scozzari, and Bruno Simeone. Political districting: from classical models to recent approaches. *Annals of Operations Research*, 204(1):271–299, 2013.

[45] Federica Ricca and Bruno Simeone. Local search algorithms for political districting. *European Journal of Operational Research*, 189(3):1409–1426, 2008.

[46] Zachary Schutzman and MGGG. The known sizes of grid metagraphs, 2018. `https://data-democracy.org/table.html`.

[47] Maral Shahmizad and Austin Buchanan. Political districting to minimize county splits. *Operations Research*, 73(2):752–774, 2025.

[48] Takeshi Shirabe. A model of contiguity for spatial unit allocation. *Geographical Analysis*, 37(1):2–16, 2005.

[49] Takeshi Shirabe. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design*, 36(6):1053–1066, 2009.

[50] Rahul Swamy, Douglas M. King, and Sheldon H. Jacobson. Multiobjective optimization for politically fair districting: a scalable multilevel approach. *Operations Research*, 71(2):536–562, 2023.

[51] US Census Bureau. State legislative districts, 2022. `https://www.census.gov/programs-surveys/decennial-census/about/rdo/state-legislative-district.html`.

[52] US Census Bureau. Congressional districts, 2023. `https://www.census.gov/programs-surveys/decennial-census/about/rdo/congressional-districts.html`.

[53] Hamidreza Validi and Austin Buchanan. A note on "A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs". *Networks*, 73(1):135–142, 2019.

[54] Hamidreza Validi and Austin Buchanan. The optimal design of low-latency virtual backbones. *INFORMS Journal on Computing*, 32(4):952–967, 2020.

[55] Hamidreza Validi and Austin Buchanan. Political districting to minimize cut edges. *Mathematical Programming Computation*, 14(4):623–672, 2022.

[56] Hamidreza Validi, Austin Buchanan, and Eugene Lykhovyd. Imposing contiguity constraints in political districting models. *Operations Research*, 70(2):867–892, 2022.

[57] Sudheendra Vijayanarasimhan and Kristen Grauman. Efficient region search for object detection. In *CVPR 2011*, pages 1401–1408. IEEE, 2011.

[58] Yiming Wang, Austin Buchanan, and Sergiy Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, 166(1-2):241–271, 2017.

[59] Justin C Williams. A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs. *Networks*, 39(1):53–60, 2002.

[60] Justin C Williams. A zero-one programming model for contiguous land acquisition. *Geographical Analysis*, 34(4):330–349, 2002.

[61] Jack Zhang, Hamidreza Validi, Austin Buchanan, and Illya V Hicks. Linear-size formulations for connected planar graph partitioning and political districting. *Optimization Letters*, 18:19–31, 2024.

[62] Andris A Zoltners and Prabhakant Sinha. Sales territory alignment: A review and model. *Management Science*, 29(11):1237–1256, 1983.