Minimizing Congestion for Balanced Dominators

Yosuke Mizutani  
yos@cs.utah.edu  
University of Utah  
Salt Lake City, Utah, USA

Annie Staker  
anne.staker@utah.edu  
University of Utah  
Salt Lake City, Utah, USA

Blair D. Sullivan  
sullivan@cs.utah.edu  
University of Utah  
Salt Lake City, Utah, USA

ABSTRACT

A primary challenge in metagenomics is reconstructing individual microbial genomes from the mixture of short fragments created by sequencing. Recent work leverages the sparsity of the assembly graph to find \( r \)-dominating sets which enable rapid approximate queries through a dominator-centric graph partition. In this paper, we consider two problems related to reducing uncertainty and improving scalability in this setting.

First, we observe that nodes with multiple closest dominators necessitate arbitrary tie-breaking in the existing pipeline. As such, we propose finding sparse dominating sets which minimize this effect via a new congestion parameter. We prove minimizing congestion is NP-hard, and give an \( \mathcal{O}(\sqrt{\Delta}) \) approximation algorithm, where \( \Delta \) is the max degree.

To improve scalability, the graph should be partitioned into uniformly sized pieces, subject to placing vertices with a closest dominator. This leads to balanced neighborhood partitioning: given an \( r \)-dominating set, find a partition into connected subgraphs with optimal uniformity so that each vertex is co-assigned with some closest dominator. Using variance of piece sizes to measure uniformity, we show this problem is NP-hard if \( r \geq 1 \). We design and analyze several algorithms, including a polynomial-time approach which is exact when \( r = 1 \) (and heuristic otherwise).

We complement our theoretical results with computational experiments on a corpus of real-world networks showing sparse dominating sets lead to more balanced neighborhood partitionings. Further, on the metagenome HuSB1, our approach maintains high query containment and similarity while reducing piece size variance.

1 INTRODUCTION

Microbial communities play a critical role in many aspects of human health (e.g., gut microbiomes) and ecosystems (e.g., marine ecology), and understanding their composition and function has been increasingly important in biological and medical research. Much of the work on these communities focuses on analyzing the genomic material (DNA and RNA) of the constituent microorganisms, a research area called metagenomics. A primary challenge in the field is reconstructing individual genomes from the mixture of short fragments created by shotgun sequencing [15].

One practical approach that has gathered significant recent attention utilizes a metagenome assembly graph to guide analyses. Commonly, this is done with a compact De Bruijn graph, or (c)DBG, where vertices correspond to DNA subsequences called \( k \)-mers and edges indicate potential compatibility in an assembly (almost complete overlap). Since the graphs corresponding to real-world metagenomic datasets may have tens of millions of vertices, scalable methods for analysis are imperative. Recent work of Brown et al. [3], implemented in spacegraphcats\(^1\), leveraged the sparsity of these graphs to enable efficient indexing and querying using partial information about suspected constituent microbes. Their approach relies on finding an \( r \)-dominating set (using Dvorak’s approximation algorithm for sparse graphs [5]), then partitioning the assembly graph into bounded-radius pieces by assigning each vertex to one of its closest dominators. The process is repeated on the piece graph to form a hierarchy of dominating sets which enables effective navigation and categorization of the data. Initial experiments demonstrated the approach can improve the completeness of partial genomes for microbes present in a community and also reveal significant strain variation in real-world microbiomes [3].

Despite these promising results, several key challenges remain. Here, we focus on those related to partitioning the metagenome assembly graph into pieces around the dominators. In particular, the current dominating set algorithm in [3] may choose dominators that are close together in the cDBG, leading to uncertainty in how a region should be “carved up” into pieces. Thus, the resulting piece graph may reflect the tie-breaking rules more than underlying ground-truth associations intrinsic to the cDBG/network. Further, the sizes of the pieces may be imbalanced, counteracting any advantage gained by using the hierarchy to prune away irrelevant regions of the cDBG.

In this work, we tackle these challenges by (a) introducing a notion of sparse dominating sets which rewards “scattering” dominators, with the aim of generating pieces which are biologically meaningful and inherently more stable, and (b) considering algorithms for balanced neighborhood partitioning: assigning vertices to dominators which minimize variation in the resulting piece sizes.

---

\(^1\)https://github.com/spacegraphcats/spacegraphcats

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’22, August 14–18, 2022, Washington, DC, USA.  
© 2022 Association for Computing Machinery.  
ACM ISBN 978-1-4503-9385-0/22/08 . . . $15.00  
https://doi.org/10.1145/3534678.3539371

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’22, August 14–18, 2022, Washington, DC, USA.  
© 2022 Association for Computing Machinery.  
ACM ISBN 978-1-4503-9385-0/22/08 . . . $15.00  
https://doi.org/10.1145/3534678.3539371
While we defer formal definitions to Section 3, our approach to avoiding nodes with multiple closest dominators is based on minimizing congestion, which measures the average number of dominators appearing in an arbitrary vertex neighborhood. We show that Minimum Congestion r-Dominating Set is NP-hard and present an $O(\sqrt{r})$-approximation algorithm running in $O(\Delta^2 n \log n)$ time, where $\Delta$ denotes the maximum degree. We compare this with the $O(r \log n)$ standard approximation algorithm for finding a (smallest) $r$-Dominating Set. We note that sparse dominating sets have no explicit size restriction, and discuss trade-offs between solution size and congestion.

Once we have an $r$-dominating set, the problem becomes one of partitioning the vertices into pieces so that (a) each vertex is assigned to a piece containing one of its closest dominators, and (b) the pieces are as equal in size as possible. For the latter condition, we minimize the variance of the piece size distribution in Balanced Neighborhood Partitioning. We show this is polynomial-time solvable when $r = 1$ and NP-hard when $r \geq 2$, even when there are only two dominators. Despite this, $Prt-Branch$ establishes the problem is fixed parameter tractable (FPT) in graphs of bounded degree when parameterized by the number of vertices equidistant from multiple dominators. Further, when $r = 1$, we give an exact $O(n^3)$ algorithm $Prt-Layer$ in general graphs using flow-based techniques; when $r \geq 2$, this yields a heuristic. Finally, we compare with a linear-time greedy heuristic, $Prt-Weight$. These algorithms are described in Section 4.

We implemented all of the above algorithms using C++ in an open-source repository and tested their performance on a large corpus of real-world networks, including a variety of metagenome graphs. Experimental results demonstrate that the choice of dominating set can significantly impact the runtime and solution quality of balanced neighborhood partitioning algorithms, with sparse sets out-performing their smaller but more congested analogues. Finally, we present preliminary results indicating that low-congestion dominating sets do not significantly degrade the fidelity of queries using partial genome bins on HuSB1, a metagenome analyzed in [3], a critical requirement for their downstream adoption.

## 2 PRELIMINARIES

### 2.1 Notation & Terminology

Given a graph $G = (V, E)$, we write $n = |G| = |V|$ for the number of vertices and $m = |G| = |E|$ for the number of edges. The distance between two vertices $x, y \in V$, denoted $d_G(x, y)$, equals the minimum number of edges in an $x, y$-path in $G$; if no such path exists, we set $d(x, y) := \infty$. The distance between a vertex $v \in V$ and vertex set $X \subseteq V$ is defined as $d_G(v, X) := \min_{x \in X} d_G(v, x)$. We use $N(v)$ and $N[v]$ to denote the open and closed neighborhoods of a vertex $v$, respectively. We write $N'[v]$ for the $r$-neighborhood: the set of all vertices at distance at most $r$ from $v$. For a vertex set $X \subseteq V$, $N'[X]$ denotes the union of $N'[x]$ for all $x \in X$. The $r$-th power of $G$ is defined as $G^r := (V, \{ uv \in E, d_G(u, v) \leq r \})$.

We write $deg_G(v) = d_G(v)$ for the degree of a vertex $v$, and $\delta$ and $\Delta$ for the minimum and maximum degree of $G$, respectively. A graph is called regular if $\delta = \Delta$. We denote the induced subgraph of $G$ on a set $X \subseteq V$ by $G[X]$. For an edge $e \in E$, we write $G/e$ for the graph formed from $G$ by contracting the edge $e$; for a vertex set $X \subseteq V$, $G/X$ denotes the graph obtained by contracting all edges in $G[X]$.

### 2.2 Dominating Sets & Related Problems

For a graph $G = (V, E)$, a vertex set $D \subseteq V$ is called a dominating set if $N[D] = V$. An $r$-dominating set generalizes this notion and is defined as a set $D \subseteq V$ such that $N'[D] = V$. The problems asking for such sets of minimum cardinality are called Minimum Dominating Set (MDS) and $r$-Dominating Set, respectively, and are NP-complete, even for regular graphs of degree 4 [8]. An $r$-perfect code is an $r$-dominating set $D$ such that every vertex $v \in V$ satisfies $|N'[v] \cap D| = 1$. It is NP-complete to determine whether a graph has a perfect code [11, 12].

The problem $r$-Dominating Set is further generalized by Minimum Weighted $r$-Dominating Set (MWDS), where each vertex has non-negative weight, and one seeks an $r$-dominating set of minimum total weight. Although MDS and MWDS cannot be approximated within a factor $c \log |V|$ for some constant $c > 0$ in polynomial time (unless $P=NP$) [1, 16], it is known that MWDS has a polynomial-time approximation scheme (PTAS) in growth-bounded graphs with bounded degree constraint [19].

### 3 SPARSE DOMINATING SETS

In this section, we formulate the problem of finding sparse dominating sets, establish its hardness, and describe several heuristics along with an integer linear programming (ILP) formulation.

To define a sparse dominating set, we first introduce the notion of congestion which measures how frequently a given set of vertices overlaps with the $r$-neighborhoods in a graph.

**Definition 3.1.** Given a graph $G = (V, E)$, vertex set $S \subseteq V$, and radius $r \in \mathbb{N}$, the $r$-congestion of $S$ at a vertex $v \in V$, denoted $cong_r(S, v)$, is $|N'[v] \cap S|$. The average $r$-congestion of $S$ in $G$ is then $\text{cong}_r(S) = \frac{1}{|S|} \sum_{v \in V} cong_r(S, v)$.

We observe that the average congestion of a given set $S$ be computed directly from the neighborhood sizes of vertices in $S$.

**Lemma 3.2.** Average congestion can be computed as $\text{cong}_r(S) = \frac{1}{|S|} \sum_{v \in S} |N'[v]|$.

We say an $r$-dominating set is sparse when it achieves low average $r$-congestion, naturally leading to the following problem.

**Minimum Congestion $r$-Dominating Set (MCDS)**

**Input:** A graph $G = (V, E)$ and radius $r \in \mathbb{N}$.

**Problem:** Find an $r$-dominating set $D \subseteq V$ such that $\text{cong}_r(D)$ is minimized.

We remark that this is distinct from the class of problems studied in [6, 10] which put uniform local constraints on each vertex (e.g. that they are dominated at least $\lambda$ and at most $\mu$ times).

We write $mac_r(G)$ (mac($G$) when $r = 1$) for the minimum average congestion attainable by any $r$-dominating set on $G$. By Lemma 3.2, $|V| \cdot \text{cong}_r(D)$ equals the weighted sum over $D$ of $w(v) = |N'[v]|$. Thus, MCDS is a specialization of MWDS. Furthermore, like other dominating set problems [18], MCDS ($G, r$) is...
In general, a minimum congestion dominating set will not also be proving their sizes can diverge arbitrarily. Further, by definition, an is hard via its relationship to perfect codes. Specifically, additionally, we observe that determining the value of mac(G) is hard via its relationship to perfect codes. Specifically, G admits an r-perfect code if and only if mac(G) = 1, but determining the existence of a perfect code is NP-hard [11, 12].

A minimum congestion dominating set thus represents the “distance” to a perfect code, leading to a natural graph editing problem whose optimal solution is bounded by a linear function of mac(G).

**Perfect Code Editing**

**Input:** A graph G = (V, E) and integer k.

**Problem:** Is there an edge set S ⊆ E of size at most k such that G’ = (V, E \ S) admits a perfect code?

**Theorem 3.4.** Let PCE(G) be the minimum k so (G, k) is a yes-instance of Perfect Code Editing. Then PCE(G) ≤ (mac(G) − 1)n ≤ 2 · PCE(G).

**Proof.** Given a dominating set D ⊆ V that attains mac(G), let v ∈ V be a vertex such that cong(D, v) > 1. Then, removing an edge uv for any u ∈ D \ {v} will decrease (mac(G) · n) by at least 1, so PCE(G) ≤ (mac(G) − 1)n. Given a perfect code D’ ⊆ V of G’, any edge addition can increase (mac(G) · n) by at most 2, so (mac(G) − 1)n ≤ 2 · PCE(G).

**3.1 Properties of Minimum Congestion Dominating Sets**

In general, a minimum congestion dominating set will not also be a minimum dominating set, and in Figure 1 we give a construction proving their sizes can diverge arbitrarily. Further, by definition, we have mac(G) ≥ 1 for any graph; we give a degree-based upper bound below.

**Theorem 3.5.** mac(G) ≤ (d + 1)/2 for every graph G, where d is the average degree of G.

**Proof.** Let D ⊆ V be a minimal dominating set (i.e. every proper subset D’ ⊂ D is not dominating). Then, D := V \ D is also a dominating set. Now, (\sum_{v \in D} N[v]) = \sum_{v \in D} (\sum_{\{u \in D : (u, v) \in E\}} 1) = \sum_{v \in D} |N[v]|. Thus, any minimum congestion dominating set must also be minimum in size. The result follows directly, since MDS is NP-hard in regular graphs [8].

**3.2 Algorithms**

We now describe several greedy algorithms along with an ILP formulation for MCDS.

**3.2.1 Greedy Algorithms.** We first recall the standard greedy algorithm for Minimum Dominating Set which we call Dom-Degree. At each step, the algorithm chooses a vertex v ∈ V such that the number of undominated vertices in N’[v] is maximized. To instead target minimizing congestion, we prioritize based on the ratio of the undominated vertices. Specifically, Dom-Ratio chooses a vertex v ∈ V such that \( \frac{|N'[v]|}{|N'[v]|} \) is maximized (equivalently, \( \frac{|N'[v]|}{|N'[v]|} \) is minimized) at each step given a partial dominating set D ⊆ V. In both algorithms, ties are broken arbitrarily. While Dom-Degree is an \( O(r \log \Delta) \)-approximation for r-Dominating Set, it is not for MCDS (Theorem 3.6).

**Theorem 3.6.** Dom-Degree does not approximate MCDS.

**Proof.** Let r = 1. Consider a graph G = (V, E) that has a biclique (A, B) with |A| = |B| = k and one attached leaf x_v for each v ∈ A \ B. Thus, n = |V| = 4k. In the first two iterations, Dom-Degree should choose vertices u ∈ A and v ∈ B. But then, all 4(k − 1) vertices in V \ {u, v, x_u, x_v} equally have one undominated vertex in their neighborhoods. Thus, Dom-Degree may choose D := A \ B, which gives cong(D) = n/8 + 1. However, D := V \ D is a perfect code, so mac(G) = 1.

In contrast, Dom-Ratio can produce sets which are arbitrarily bigger than the minimum dominating set. (Figure 1 (right) is an example of this), yet we prove it is an approximation for MCDS:

**Theorem 3.7.** Dom-Ratio is an \( O(\sqrt{N}) \)-approximation algorithm for MCDS.

To evaluate smarter tie-breaking strategies, we define Dom-Degree-Plus to be Dom-Degree with ties broken using Dom-Ratio’s criteria (the ratio of undominated vertices), and Dom-Ratio-Plus analogously using Dom-Degree’s criteria (the number of undominated vertices). Further ties are randomly broken. All of these algorithms run in time \( O(\Delta^3 n \log n) \) (details in Appendix A.3).
3.2.2 Integer Programming. We observe that one may obtain optimal solutions to r-DOMINATING SET and MCDS using an ILP solver, allowing empirical evaluation of approximation ratios in Section 5. We use the following ILP formulation for r-DOMINATING SET:

\[
\begin{align*}
\text{Dom-MDS} \\
\text{Let } x_v \in \{0, 1\} \text{ be variables for all } v \in V. \\
\text{Minimize } \sum_{v \in V} x_v \\
\text{Subject to } \sum_{w \in N^*(v)} x_w \geq 1 \text{ for all } v \in V.
\end{align*}
\]

Similarly, we formulate MCDS as follows:

\[
\begin{align*}
\text{Dom-MAC} \\
\text{Let } x_v \in \{0, 1\} \text{ be variables for all } v \in V. \\
\text{Minimize } \sum_{v \in V} |N^*(v)| x_v \\
\text{Subject to } \sum_{w \in N^*(v)} x_w \geq 1 \text{ for all } v \in V.
\end{align*}
\]

This is based on the fact that MCDS is a specialization of MINIMUM WEIGHTED r-DOMINATING SET. In both cases, the solution set is provided by \( \{ v \in V : x_v = 1 \} \).

4 NEIGHBORHOOD PARTITIONING

We now turn to the second problem arising in our metagenomics application: partitioning the vertex set into pieces around a set of r-dominators as evenly as possible. We first formalize the notion of a neighborhood partitioning and use variance to define its balance. After establishing that the resulting problem is NP-hard (and remains so under very restrictive conditions), we show that for radius 1, a flow-based approach gives a polynomial-time solution. Finally, we describe several algorithmic approaches for obtaining both exact and heuristic solutions.

We begin by considering the more general setting where we are given \( G = (V, E) \) and landmarks \( L = \{u_1, \ldots, u_t\} \subseteq V \) to partition into \( |L| \) disjoint sets so that (a) each piece contains exactly one landmark, (b) every vertex is assigned to a piece with one of its closest landmarks from \( L \), and (c) for every piece \( A \), the induced subgraph \( G[A] \) preserves the distance between the landmark and other vertices in \( A \).

**Definition 4.1.** Given a graph \( G = (V, E) \) and landmarks \( L = \{u_1, \ldots, u_t\} \subseteq V \), we say \( \mathcal{A} = \{A_1, \ldots, A_t\} \) is a neighborhood partitioning of \( G \) with respect to \( L \) if and only if \( \mathcal{A} \) is a set partition of \( V \) and \( d_{G[A_i]}(v, u_i) = d_{G}(v, L) \) for every \( 1 \leq i \leq t, v \in A_i \).

We note that if \( L \) is an r-dominating set of \( G \), the resulting pieces will necessarily have radius at most \( r \), making \( G/\mathcal{A} \) an r-shallow minor. This is essential for spacegraphcats to maintain efficiency guarantees when computing the dominating sets in their hierarchy, since it ensures graphs remain within the assumed sparse class (bounded expansion) [3].

We now define the problem of finding a set of pieces whose size distribution is as even as possible.

**Balanced Neighborhood Partitioning (BNPart)**

**Input:** A graph \( G = (V, E) \) and landmarks \( L = \{u_1, \ldots, u_t\} \subseteq V \)

**Problem:** Find a neighborhood partitioning \( \mathcal{A} = \{A_1, \ldots, A_t\} \) of \( G \) on \( L \) such that the piece-size population variance \( \text{Var}(\{|A_i| : A \in \mathcal{A}\}) \) is minimized.

A key observation is that since \( V \) and \( L \) are given to us, the average size of \( \mathcal{A} \) is \(|V|/|L|\), which is fixed. This results in the following equivalence:

**Theorem 4.2.** A neighborhood partitioning \( \mathcal{A} \) gives the minimum piece-size variance if and only if the square sum of the pieces-sizes is minimized.

We now establish the hardness of BNPart.

**Theorem 4.3.** BNPart is NP-hard. It remains hard even if \( \max_{v \in V} d(v, L) = 2 \) or if \( |L| = 2 \).

We break the proof of Theorem 4.3 into two lemmas, first showing the case when the landmarks are a 2-dominating-set by reduction from Exact cover by 3-sets.

**Exact Cover by 3-sets (X3C)**

**Input:** Set \( X \) with \(|X| = 3q\) and a collection \( C \) of 3-element subsets of \( X \).

**Problem:** Is there a collection \( C' \subseteq C \) such that every element of \( X \) occurs in exactly one member of \( C' \)?

**Lemma 4.4.** BNPart is NP-hard when the landmarks are distance at most two from all vertices.

**Proof.** Consider an instance of X3C with \( n = |C| \). If \( n < q \), then output “no”. Assuming \( n \geq q \), we construct a graph \( G = (V, E) \) as follows. The set of vertices \( V \) includes \( n \) landmarks \( L = \{u_i | 1 \leq i \leq q\} \), \( C \), \( X \), and attached leaves; 3 leaves are attached to each of the \( n - q \) vertices in \( L \). The edges are constructed as follows: \( L \) and \( C \) form a biclique; for each \( C \in C \), \( C \) is connected to all elements of \( X \) it contains. This construction can be done in time \( O(n^3) \). See Figure 2 for an example.

We return “yes” if and only if BNPart returns a partition with pieces of equal size (by construction, this must be 5). Let \( L' \subseteq L \) be the landmarks that do not have attached leaves. If the given X3C instance admits an exact cover \( C' \), then for each \( C \in C' \), we assign one landmark in \( L' \) to \( C \) and all elements of \( C \). Then, we assign each of \( L - L' \) to \( n - q \) unused sets in \( C \), resulting in pieces of equal sizes.

Conversely, suppose BNPart returns an equally-sized partition. Because each piece must have 5 vertices, \( n - q \) sets in \( C \) are assigned to \( L \subseteq L' \). Also, since \(|L'| = q\), each landmark in \( L' \) must have exactly one vertex of \( C \) in its piece. Then, each piece contains exactly 3 elements in \( X \), and this is an exact cover by 3-sets.

**Lemma 4.5.** BNPart is NP-hard when there are two landmarks.

**Proof.** We reduce from SATISFIABILITY (SAT). Let \( x_1, \ldots, x_m \) be the variables and \( \phi_1, \ldots, \phi_m \) be the clauses appearing in a SAT
We now present several algorithms for computing neighborhood dominating set. Showing an example: \( C_1 = \{x_1, x_3, x_6\}, C_2 = \{x_1, x_2, x_3\}, C_3 = \{x_2, x_3, x_4\} \) with solution \( \{C_1, C_3\} \). Landmarks are a 2-dominating set.

instance. We construct a graph \( G = (V, E) \) as follows. The set of vertices \( V \) includes two landmarks \( L = \{u_1, u_2\} \), 2n vertices \( X = \{x_1, x_2, \ldots, x_n\} \) representing SAT literals, \( n \) variables \( Y = \{y_1, \ldots, y_n\} \), \( m \) clauses \( \Phi = \{\phi_1, \ldots, \phi_m\} \), and attached leaves. A total of \( n - 1 \) leaves are attached to each of \( Y \) and \( \Phi \), and \( n(n + m) \) leaves are attached to \( u_2 \). The set of edges \( E \) is constructed as follows: \( L \) and \( X \) form a biclique; each \( y_i \in Y \) is connected to \( x_i \) and \( x_i \); each \( \phi_j \in \Phi \) is connected to all literals in clause \( \phi_j \). This construction can be done in time \( O(n(n + m)) \). See Figure 3 for an example.

Let \( \{A, B\} \) be a partition such that \( u_1 \in A \) and \( u_2 \in B \). Note that in any neighborhood partitioning, each piece must be connected, so \( B \) must contain all \( n(n + m) \) of the leaves attached to it. Likewise, if a piece includes \( Y_1 \) or \( \Phi_j \), it must also include their attached leaves.

We return ”yes“ if \( \text{BNPart} \) returns a partition of equal sizes. If the given SAT instance is satisfiable, then we include \( Y_1 \) and \( \Phi_j \), and all true literals of \( SAT \).

Definition 4.7 (Neighborhood Kernel). A graph \( G = (V, E) \) and landmarks \( L \subseteq V \), a neighborhood kernel \( H \) is a digraph with vertex set \( V \) and edge set \( E' = \{(v, w) : \text{ov} \in E, \ d(v, L) + 1 = d(w, L)\} \).

By construction, for every non-landmark vertex \( v \) in a neighborhood kernel, all shortest paths from \( v \) to any landmark must include one of \( v \)'s in-neighbors in \( H \), and thus \( v \) must be in the same piece as one of its in-neighbors. If \( v \) has only one in-neighbor \( w \), then \( v \) and \( w \) must be assigned to the same piece. We encapsulate this idea in the following data reduction, noting that by definition, all landmarks are kept in the compact neighborhood kernel.

Definition 4.8 (Compact Neighborhood Kernel). A graph \( G = (V, E) \) and landmarks \( L \subseteq V \), a compact neighborhood kernel \( (H_c, \phi) \) is a pair consisting of a digraph \( H_c = (V_c, E_c) : H/\mathcal{A} \) and a map \( \phi : V_c \rightarrow 2^{V} \) defining its "bags", where the collection of bags \( \mathcal{A} := \{\phi(v) : v \in V_c\} \) is a partition of \( V \). Additionally, we require the following conditions:

- Each vertex \( v \in V_c \) is the representative for its bag, and must be the closest to \( L \) in \( G \) among \( \phi(v) \).
- All bag members must be assigned to the same landmark in any valid neighborhood partitioning.
Figure 4: Visualization of neighborhood kernels. Given a graph $G$ with three landmarks $a, b, c$ (left), the neighborhood kernel $H$ of $G$ (center) identifies groups dotted in red which must all be assigned to the same landmark. At right, the compact neighborhood kernel of $G$ has one representative vertex (solid border) per bag; colors indicate an optimally balanced neighborhood partitioning.

\begin{itemize}
  \item $\phi(v)$ is maximal subject to these conditions.
\end{itemize}

We visualize this process in Figure 4. In Appendix A.3, we describe algorithms (Nbr–Kernel1 and Compact–Nbr–Kernel1) for creating (compact) neighborhood kernels in $O(n + m)$ time. Finally, we note that the maximum degree cannot increase under these transformations (Lemma A.4). We use this fact to bound the running time of our branch-and-bound algorithm (Section 4.1.3).

4.1.2 Heuristic Algorithms. We developed two heuristics for BNPart. Prt-Weight is a linear-time $O(n + m)$ greedy algorithm that works on the compact neighborhood kernel. From landmarks, it traverses all bags in BFS order, assigning each bag to the smallest piece among viable candidates.

**Theorem 4.9.** Prt-Weight gives a valid neighborhood partitioning in $O(n + m)$ time.

**Proof.** We proceed by induction on distance from a landmark. As the base case, landmarks assign themselves and their bag members, which is the only valid option. For the inductive step, consider when $v$ is assigned. Because bags are processed in BFS order, $v$’s in-neighbors must have already been assigned to a valid landmark. Thus, extending the current assignment with any choice at $v$ meets all requirements, and Prt-Weight produces a valid partitioning.

The running time is $O(n + m)$ because this algorithm performs BFS, and a landmark’s piece size can be updated in $O(1)$.

Prt-Layer is a polynomial-time $O(|L| n^3)$ algorithm that is exact when $L$ is a (1-)dominating set and heuristic otherwise. It works on the neighborhood kernel and solves SBA at each layer starting from the one closest to landmarks (a layer $V_i$ is the set of vertices such that the distance to the closest landmark is $i$).

**Theorem 4.10.** Prt-Layer gives a neighborhood partitioning in $O(|L| n^3)$ time. Further, it gives an exact solution to BNPart when $L$ is a dominating set.

**Proof.** If each assigned piece is connected in the neighborhood kernel, then it is a valid neighborhood partitioning because vertices $V_i$ at layer $i$ are distance $i$ away from their closest landmarks, both in the original graph and their assigned piece. The algorithm examines vertices in level order emanating from the landmarks, and by the formulation of SBA, each of the resulting pieces must be connected.

Also, from Theorem 4.6, this algorithm gives an exact solution when $L$ is a dominating set.

The running time (other than constructing a neighborhood kernel) is the sum of the time taken for SBA at each layer. At layer $i$, only the vertices in $V_i$ have choices (all vertices in earlier layers already have an assigned landmark). In a flow problem, we can efficiently preprocess already assigned vertices, thus resulting in time $O(|L| |V_i|^3)$. Because of the fact $V = \bigcup_i V_i$, the total running time is $\sum_i O(|L| |V_i|^3) = O(|L| n^3)$.

4.1.3 Branch-and-bound Algorithm. Before elaborating on our exact algorithm for BNPart, we show the following motivating result.

**Theorem 4.11.** BNPart is fixed parameter tractable (FPT) in graphs of bounded degree parameterized by $k$, the number of vertices equidistant from multiple landmarks.

**Proof.** Given a graph $G = (V, E)$, landmarks $L \subseteq V$, and their compact neighborhood kernel $(H_c, \phi)$, let $\Delta$ be the maximum degree of $G$. Let $X$ be $V(H_c \setminus L)$. Since all vertices in $X$ are equidistant from multiple landmarks, $|X| \leq k$. From Lemma A.4, each vertex in $H_c$ has at most $\Delta$ in-neighbors. Every vertex in $X$ must be in the same piece as one of its in-neighbors for a valid neighborhood partitioning, giving at most $\Delta^k$ possible assignments. Other computations can be done in polynomial time in $n$, so the brute-force approach results in time $O(\Delta^k n^{O(1)})$.

The algorithm Prt-Branch (documented and implemented in [13]) reinforces this idea by combining efficient base-case handling with naive branch-cut functionality. For a base case, we apply Prt-Layer if possible; specifically, if there is only one layer $V_1$ left and all bags are of size 1, Prt-Layer gives an exact solution in $O(|L| n^3)$ time. To obtain a lower-bound for a branch cut, we exploit the fact that our partial solutions $\mathcal{A}'$ satisfy the following property: any solution $\mathcal{A}$ extending $\mathcal{A}'$ has $\sum_{A \in \mathcal{A}} |A|^2 \geq \left(\sum_{A' \in \mathcal{A}'} |A'|^2\right) + (n - |\mathcal{A}'|)^2 / |L|$. The total running time of this algorithm is $O(\Delta^k |L| n^3)$.

4.1.4 Quadratic Programming. Finally, we give a quadratic programming formulation. Given a graph $G = (V, E)$ and landmarks $L \subseteq V$, let $(H_c = (V_c, E_c), \phi)$ be the compact neighborhood kernel.
of $G$ on $L$. Let $x_{u,v} \in \{0,1\}$ be variables for all $u \in L$, $v \in V_G$; set $x_{u,v} = 1$ if and only if $v$ is assigned to landmark $u$.

\[
\begin{align*}
\text{Minimize} & \quad \sum_{u \in L} \left( \sum_{v \in V_G} |\phi(v)| x_{u,v} \right)^2 \\
\text{Subject to} & \quad \sum_{u \in L} x_{u,v} = 1 \quad \text{for all } v \in V_G, \\
& \quad \sum_{w \in N_{G_c}(v)} x_{u,w} \geq x_{u,v} \quad \text{for all } u \in L, v \in V_G \setminus L.
\end{align*}
\]

By Theorem 4.2, the objective function guarantees minimum variance. The first constraint enforces that every vertex must be assigned to exactly one landmark, and the second constraint guarantees that every non-landmark must be in the same piece as one of its in-neighbors. The balanced neighborhood partitioning is given by $A = (A_1, \ldots, A_l)$ where $A_u = \bigcup_{v \in V_G : x_{u,v} = 1} \phi(v)$.

\section{Experiments}

To complement our theoretical results, we implemented and evaluated our algorithms on a diverse corpus of networks including cDBGs constructed from real metagenomes, small instances from DIMACS10 [2], and graphs from [14]. We categorized each graph as small, medium, or large based on the number of edges; all data, along with a summary table of statistics are available at [13]. All algorithms were implemented in C++ and experiments were conducted on a Linux machine (details in Appendix B).

\subsection{Sparse Dominating Sets}

To evaluate our algorithms for finding a low-congestion dominating set, we first tested the effectiveness of tie-breaking strategies by running Dom-Degree (denoted deg), Dom-Ratio (ratio), Dom-Degree-Plus (deg+) as well as Dom-Ratio-Plus (ratio+) on small instances. The algorithms with a tie-breaking strategy consistently outperformed those without one, finding smaller dominating sets in 78\% of the experiments and less-congested dominating sets in 93\% of them. Running time was also improved by tie-breaking strategies in 55\% of instances, likely due to its proportionality to solution size.

Based on these findings, we restricted our attention to deg+ and ratio+ for the remaining experiments. We evaluated them, along with Dom-SGC (sgc-d), Dom-MDS (mds), and Dom-MAC (mac) on the full corpus with radii $[1, 2]$ (plus radii $[3, 5, 10]$ on small and medium instances). We used a default 3-hour timeout, reduced to 1 hour for mds and mac for radius $> 1$ and medium/large instances.

Figure 5 (left) shows the distribution of running times sorted by $||G||$, the number of edges in the $r$-th power of $G$. We observe a linear trend for sgc-d, deg+, and ratio+, establishing efficiency. While the exact algorithms mds and mac are prone to timing out, they did finish on some larger instances. We hypothesize this success is directly related to reduction rules in the ILP solver related to vertices with degree 1 and 2.

To evaluate solution quality, we plotted the relationship between the solution size and average congestion, both relative to the best-known (Figure 5, right). As might be expected, deg+ and mds find smaller, more congested dominating sets, while ratio+ and mac find larger, sparser ones. The algorithm used in the prior metagenomic analysis (sgc-d) intermediates between the two, but often at the cost of being much larger or more congested.

\subsection{Balanced Neighborhood Partitioning}

Turning to the problem of generating uniformly-sized pieces around a set of dominators, we tested the scalability and solution quality of the algorithms for BNPart as well as the impact of choosing smaller versus sparser dominating sets. To this end, we ran Prt-SGC (sgc-p), Prt-Weight (wt), Prt-Layer (layer), Prt-Branch (branch), and Prt-QP (qp) using dominating sets produced by sgc-d, deg+, and ratio+. All runs were subject to a 1-hour timeout.

We first evaluated the running time and runtime ratio for all BNPart algorithms (Table 1) on the corpus of small instances. The algorithm wt was consistently fastest, followed by sgc-p; both had no timeouts. The polynomial-time layer was slower than the linear-time approaches but still completed nearly all small and medium instances. It is particularly notable that when $r = 1$, it successfully output optimal solutions, which qp could not find. We also observed that qp’s performance improves when given a cDBG and dominators from deg+ or ratio+. Lastly, branch was unable to finish with larger radii, even on these small instances, eliminating it from use in additional experiments.
5.3 Metagenome Neighborhood Queries

Given the motivation for this work is improving the metagenomics analysis pipeline from [3], we also assessed the impact of our techniques in this setting. Specifically, we verified that using sparse dominating sets and balanced neighborhood partitionings from our algorithms does not significantly degrade the containment and similarity of neighborhood queries in HuSB1, a large real-world metagenome [9]. We used Brown et al.’s replication pipeline6 and compared our algorithms (deg+ and ratio+ for MCDS; wt for BNPart) to those used in [3] (sgc-d and sgc-p, respectively). While [3] restricted their attention to radius 1, we also include results from radius 2, as this is of interest in ongoing related work by Brown et al.

Before running neighborhood queries, we evaluated the piece size distributions as shown in Figure 7 (left, center). We observe that independent of dominator selection, wt successfully reduced the piece size variance. When comparing dominating sets, we see a trend consistent with other experiments. sgc-d chooses many dominators, resulting in smaller more uniform piece sizes. deg+ has the fewest dominators and higher variance, larger pieces and ratio+ gives larger, balanced pieces.

Finally, we re-ran the neighborhood queries from [3]. While the containment was completely preserved, the similarities7 exhibited mild variation, shown in Figure 7 (right). While there is a slight overall reduction (partly explainable by having fewer pieces), the difference is marginal at $r = 2$. These preliminary results indicate that our techniques can significantly improve the balance of piece sizes (leading to more efficient downstream analysis when a hierarchy of dominating sets is constructed) without significantly degrading fidelity of queries. Additional experiments are needed to better assess the impact of this work in the metagenomic setting.

6 CONCLUSIONS & FUTURE WORK

This paper tackles two problems arising in a recent approach to metagenome analysis: finding ($r$-)dominating sets which minimize the number of vertices with multiple closest dominators, and partitioning the vertices of a graph into connected pieces around a set of landmarks minimizing variance of the piece sizes while guaranteeing every vertex is assigned to some closest landmark.

We formalize the first using the notion of congestion, and show that finding minimum congestion $r$-dominating sets (MCDS) is NP-hard. We introduce linear-time algorithms for finding low-congestion dominating sets and evaluate their effectiveness on a large corpus of real-world data, showing trade-offs between solution size and average congestion. It remains open whether there is a constant upper bound on the minimum average congestion (the largest value observed in our experiments was 3.7351); further,
we believe the approximation bound on MCDS is not tight. We are intrigued by the connection of this problem to Perfect Code Editing; this could be a fruitful direction for future work.

Turning to the partitioning problem (BNPART), we show that at radius 1, flow-based techniques give exact solutions in polynomial time, but the problem becomes NP-hard even in very restricted cases as soon as $r \geq 2$. Our heuristics, however, produce nearly optimal results in our experiments. Further, we show that using sparse dominating sets (such as those from star 10+) improves both the running time and solution quality of algorithms for BNPART. A natural question is whether there are alternative measures of partition balance in real-world data which avoid the inverse relationship between dominating set size and the variance of the piece size distribution.

Finally, we integrate our algorithms into the metagenomic analysis pipeline used in [3] and demonstrate that on a large real metagenome (HuSB1), sparse dominating sets and balanced neighborhood partitionings reduce piece size variability relative to the prior approaches without significantly degrading the fidelity of neighborhood queries. It remains to see how these preliminary results extend to other metagenomic datasets and queries.

ACKNOWLEDGMENTS

Special thanks to Rachel Walker for early discussions on notions of sparse dominating sets & related terminology, and to Titus Brown & Taylor Reiter for assistance with metagenomic datasets and the spacegraphcats software. This work was supported by grant GBMF4560 to Blair D. Sullivan from the Gordon and Betty Moore Foundation.

REFERENCES

[1] Noga Alon, Dana Moshkovitz, and Shmuel Safra. 2006. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)* 2, 2 (2006), 153–177.

[2] David A Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 2013. Graph partitioning and graph clustering. Vol. 588. American Mathematical Society, Providence, RI.

[3] C. Titus Brown, Dominik Moritz, Michael P. O’Brien, Felix Reidl, Taylor Reiter, and Blair D. Sullivan. 2020. Exploring neighborhoods in large metagenome assembly graphs using spacegraphcats reveals hidden sequence diversity. *Genome Biology* 21, 1 (06 Jul 2020), 164.

[4] Rayan Chikhi, Antoine Limasset, and Paul Medvedev. 2016. Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* 32, 12 (06 2016), i201–i208.

[5] Zdeněk Dvořák. 2013. Constant-factor approximation of the domination number in sparse graphs. *European Journal of Combinatorics* 34, 5 (2013), 833–840.

[6] Carl Eduardson and Felix Reidl. 2020. A General Kernelization Technique for Domination and Independence Problems in Sparse Classes. In 15th International Symposium on Parameterized and Exact Computation (IPEC 2020) (Leibniz International Proceedings in Informatics (LIPIcs)), Vol. 180). 11:1–11:15.

[7] Lester Randolph Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* 8 (1956), 399–404.

[8] Michael R Garey and David S Johnson. 1979. *Computers and intractability*. Vol. 174. freeman San Francisco.

[9] Ping Hu, Lauren Tom, Andrea Singh, Brian C. Thomas, Brett J. Baker, Yvette M. Piceno, Gary L. Andersen, Jillian F. Banfield, and Nicole Dubilier. 2016. Genome-Resolved Metagenomic Analysis Reveals Roles for Candidate Phyla and Other Microbial Community Members in Biogeochemical Transformations in Oil Reservoirs. *mBio* 7, 1 (2016), e01669–15.

[10] Lars Jaffke, O Jougw Kong, Torstein J.F. Stromme, and Jan Arne Telle. 2019. Mimi-width III. Graph powers and generalized distance domination problems. *Theoretical Computer Science* 796 (2019), 216–236.

[11] Jan Kratochvíl. 1994. Regular codes in regular graphs are difficult. *Discrete Mathematics* 133, 1 (1994), 191–205.

[12] Marilynn Livingston and Q. Stout. 1997. Perfect Dominating Sets. *Congressus Numerantium* 79 (08 1997).

[13] Yosuke Muzutani, Annie Staker, and Blair D. Sullivan. 2022. Accompanying source code. https://github.com/TheoryInPractice/sparsedomsets.

[14] Minh Nguyen, Minh H. Diep Nguyen, and The Tran. 2020. Solving the k-dominating set problem on very large-scale networks. *Comp. Social Networks* 7 (07 2020).

[15] Christopher Quince, Alan W Walker, Jared T Simpson, Nicholas J Loman, and Nicola Segata. 2017. Shotgun metagenomics, from sampling to analysis. *Nature biotechnology* 35, 9 (2017), 833–844.

[16] Ran-Raz and Shmuel Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 475–484.

[17] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In AAAI.

[18] Peter J. Slater. 1976. R-Domination in Graphs. *J. ACM* 23, 3 (July 1976), 446–450.

[19] Zhong Wang, Wei Wang, Joon-Mo Kim, Bhavani Thuraisingham, and Wei Li Wu. 2012. PTAS for the minimum weighted dominating set in growth bounded graphs. *Journal of Global Optimization* 54, 3 (2012), 641–648.

Figure 7: Experiment results of neighborhood queries. The left panel shows log-scale distributions of the resulting piece sizes for radius 1 (top) and 2 (bottom). The right panel shows the Jaccard similarity of the queries and their neighborhoods.
A THEORETICAL RESULTS

A.1 Neighborhood Partitioning Proofs

First, we prove Theorem 4.2, which relates the minimum variance to the minimum sum of squares.

Proof of Theorem 4.2. Let \( X = \{x_1, \ldots, x_T \} \) be the piece sizes of \( \mathcal{A} \). By definition of population variance, \( \text{Var}(X) = \frac{1}{|X|} \sum_{i=1}^T (x_i - \mu)^2 = \left( \frac{1}{|X|} \sum_{i=1}^T x_i^2 \right) - \mu^2 \), where \( \mu = \frac{1}{|X|} \sum_{i=1}^T x_i = |V|/t \). Since \( \mu \) is independent of \( \mathcal{A} \), the minimum variance is always achieved with the minimum square sum. □

Next, we define Size Balanced Assignment, which can be viewed as a generalization of an assignment problem where every agent is required to have the same number of tasks.

\begin{itemize}
  \item **Problem:** \( \text{Input: } A, T, \text{ tasks, and a binary relation } R \subseteq A \times T. \)
  \item **Problem:** \( \text{Find a function } \phi : T \to A \text{ such that } (\phi(t), t) \in R \text{ for every } t \in T, \text{ and the population variance of the number of assigned tasks for each agent } \text{Var}(|\phi^{-1}(a)| : a \in A) \text{ is minimized.} \)
\end{itemize}

We use the following results to prove Theorem 4.6.

Theorem A.1. SBA can be solved in time \( O(|A|^2|T|^2 + |A||T|^3) \).

Proof. The same argument as Theorem 4.2 applies here, and SBA is equivalent to asking for an assignment minimizing the square sum of \( |\phi^{-1}(a)| \) for each \( a \in A \).

We transform this problem to the minimum-cost maximum flow problem with integral flows. Construct a multi-digraph \( G = (V, E) \). The set of vertices \( V \) includes \( A, T, \) the source \( \alpha \) and sink \( \beta \) of the flow. We construct the edges \( E \) as follows. \( A \) and \( T \) preserve their relation \( R \) as edges, oriented from \( A \) to \( T \). For every task \( t \in T \), add one edge from \( t \) to \( \beta \). For every agent \( a \in A \), add multiple edges from \( a \) to \( \alpha \) so that the number of incoming edges matches the number of outgoing edges at \( a \). Only these edges from \( a \) have costs. For the \( t \)-th edge to each \( a \), denoted \( e_{a,t} \), we set the cost to \( 2t - 1 \). All edges have capacity \( 1 \). There are at most \( 2|R| + |T| \) edges in this graph, so the construction can be done in \( O(|A||T|) \) time. See Figure 8 for an example.

Let \( c(e) = 1 \) be the capacity and \( w(e) \) be the cost of each edge. Also, let \( \hat{w} \) be the minimum cost and \( \hat{f} \) be the maximum flow amount of \((G, c, w)\) from \( \alpha \) to \( \beta \). We claim that for any \( k \in \mathbb{Z} \), we get \( \hat{w} \leq k \) and \( \hat{f} = |T| \) if and only if the SBA instance \((A, T, R)\) admits a valid assignment \( \phi : T \to A \) with the square sum of the number of assigned tasks \( \sum_{a \in A} |\phi^{-1}(a)|^2 \leq k \).

To prove the correctness in the forward direction, suppose that there exists a flow of amount \( |T| \) and cost \( \hat{w} \leq k \). The flow at \( t \in T \) is 1, and there is incoming flow from only one agent \( a \in A \), which corresponds to assignment \( t \mapsto a \). Likewise, the flow at \( a \in A \) must equal \( |\phi^{-1}(a)| \). The minimum cost can be achieved by \( \sum_{a \in A} |\phi^{-1}(a)|^2 = (2t - 1) = \sum_{a \in A} |\phi^{-1}(a)|^2 \leq k \).

On the other hand, assume that there exists a valid assignment \( \phi : T \to A \). Then, we can construct the following flow \( f : E \to \mathbb{Z} \) with amount \( |T| \) and cost \( k \). Thus, \( \hat{w} \leq k \), as desired.

\[
\begin{align*}
f(e_{a,t}) &= \begin{cases} 1 & \text{if } t \leq |\phi^{-1}(a)| \\ 0 & \text{otherwise} \end{cases} \\
f((a,t)) &= \begin{cases} 1 & \text{if } a = \phi(t) \\ 0 & \text{otherwise} \end{cases} \quad \text{for every } a \in A \\
f((t,\beta)) &= 1 \quad \text{for every } t \in T
\end{align*}
\]

Since the capacities are integral, there exists an integral maximum flow \([7]\). The Bellman-Ford algorithm can find an integer-valued minimum-cost maximum flow in time \( O(|f_{\text{max}}||V||E|) \). In our case, that is \( O(|T|(|A|+|T|)|A||T|) = O(|A|^2|T|^2 + |A||T|^3) \). □

A.2 Dom-Ratio Approximation Guarantee

In the interest of space, we sketch here only the main ideas of the proof of Theorem 3.7, which guarantees an \( O(\sqrt{|A|}) \)-approximation.

Proof Sketch of Theorem 3.7. First, observe that Dom-Ratio behaves in the same way on \((G', 1)\) as on \((G, r)\). Since \( \Delta(G') \leq (\Delta(G))^2 \), it suffices to consider \( r = 1 \).

Consider a partial dominating set \( D \subseteq V \) and chosen vertex \( v \in V \) during an iteration of Dom-Ratio. Let us call \( t = |N[D]|/|V| \) the global domination ratio, and \( \rho(v) = |N[v]|/|N[D]|/|N[v]| \) the local domination ratio. By definition, \( \rho \) has the minimum local domination ratio among all vertices at this point.

A key idea is to consider the minimum possible \( t \) for the given \( \rho \). Special cases lie in \( 0 \leq t \leq \frac{2}{3t+1} \) and \( \frac{2}{3t+1} \leq t \leq 1 \), but they have constant effects on the approximation ratio, so we omit them here.

Now, we may assume \( 0 < \rho < 1 \). Let us partition \( V \) into three parts: \( D, U = N[D] \setminus D \) and \( W := V \setminus N[D] \). Then, every vertex in \( U \) may have at most \((1 - \rho)(\Delta + 1)\) neighbors in \( W \), and every vertex in \( W \) must have at least \( \rho / (1 - \rho) \) neighbors in \( U \), which leads to the relation: \( |W|/|U| \leq (\Delta + 1)(1 - \rho)^2 / \rho \). Consequently, we get:

\[
t \geq f(r) \quad \text{and} \quad r \leq 1 - f^{-1}(t), \quad \text{where } f(x) = 1/(1 + \Delta(1 - x)^2 / x).
\]
Next, we relate \( \text{cong}(D) \) to \( t \) by the fact that every time we add a vertex \( v \) to \( D \), \( \text{cong}(D) \) increases by \( \frac{(\deg(v) + 1)}{n} \) and \( t \) increases by \( (1 - \frac{1}{n}) \frac{(\deg(v) + 1)}{n} \). Finally, we obtain: \( \text{cong}(D) \leq O(1) + \int_{2/(\sqrt{2}+1)}^{1} (1/t) \, dt \), which evaluates to \( \frac{2}{\sqrt{2}} \sqrt{\Delta} + O(1) \). Since \( t \leq \max(G) \leq \text{cong}(D) \) for all \( G \), the approximation ratio is bounded by \( O(\sqrt{\Delta}) \) when \( r = 1 \), and \( O(\sqrt{\Delta^r}) \) for general \( r \). \( \square \)

### A.3 Algorithm Details

First, we give the running time of the algorithms from Section 3.

**Theorem A.2.** Dom-Degree, Dom-Degree-Plus, Dom-Ratio, and Dom-Ratio-Plus find an \( r \)-dominating set in time \( O(2^r n \log n) \).

**Proof of Theorem A.2.** All of these algorithms continue to choose vertices until the entire graph becomes dominated, so the resulting set is an \( r \)-dominating set.

The running time is also the same among these algorithms. Consider a max heap of \( n \) elements, which stores values of the greedy criteria for each \( v \) in \( V \). We can find and remove the maximum element in \( O(\log n) \) time. After adding a vertex \( v \) to a partial dominating set \( D \), we need to update at most \( 2^r |\mathcal{V}| \) elements in the heap because all criteria are based on the number of undominated vertices in the \( r \)-neighborhood. There are at most \( n \) iterations, and since \( |\mathcal{N}^x_G| \leq 2^r \), the total running time is \( O(2^r n \log n) \). \( \square \)

Next, we show that the (compact) neighborhood kernel can be found in linear time.

**Algorithm 1: Nbr-Kernel \((G, L)\)**

**Input:** Graph \( G = (V, E) \), landmarks \( L \subseteq V \)

**Output:** Neighborhood kernel \( H \)

1. \( E' \leftarrow \emptyset \)
2. For \( e \in E \) do
   3. \( \text{cw} \leftarrow e \) such that \( d(v, L) \leq d(w, L) \) for all \( v, w \in L \)
   4. If \( d(v, L) + 1 = d(w, L) \) then
      5. \( E' \leftarrow E' \cup \{vw\} \)
3. Return \( H = (V, E') \)

**Algorithm 2: Compact-Nbr-Kernel \((G, L)\)**

**Input:** Graph \( G = (V, E) \), landmarks \( L \subseteq V \)

**Output:** Compact neighborhood kernel \( (H_c, \psi) \)

1. \( H \leftarrow \text{Nbr-Kernel}(G, L) \)
2. Initialize \( \psi : V \rightarrow V \) to the identity map
3. For \( v \in V \) sorted by ascending \( d_G(v, L) \) do
   4. If \( \deg_H(v) = 1 \) then
      5. Let \( w \in N_H^r(v) \) // only one in-neighbor
      6. Set \( \psi(v) = w \) // \( w \) belongs to bag \( w \)
   7. \( H \leftarrow H/\psi^{-1} \) // contract edge \( \psi^{-1} \)
5. Return \( (H, \psi^{-1}) \)

**Theorem A.3.** Nbr-Kernel (Compact-Nbr-Kernel) correctly gives the (compact) neighborhood kernel of the graph on the given landmarks in \( O(n + m) \) time.

**Proof.** We begin with Nbr-Kernel, where the check in Line 4 enforces Definition 4.7, ensuring correctness. For every vertex \( v \in V \), the distance from the closest landmark can be computed by BFS from \( L \) in \( O(n + m) \) time. The algorithm checks all edges in \( O(m) \), resulting in \( O(n + 2m) = O(n + m) \).

Turning to Compact-Nbr-Kernel, we traverse all vertices in BFS order from the landmarks, which guarantees each bag’s representative to be the closest to \( L \). By the definition of BNPart, Line 4 ensures that all bag members must be assigned to the same landmark because a vertex must follow its in-neighbor’s assignment. Lastly, \( \phi(v) \) is maximal for every \( v \in V \), because otherwise, there exists a vertex \( x \in V \), with at least two in-neighbors \( y, z \) such that \( v = \psi(y) = \psi(z) \neq \psi(x) \). This cannot happen because the algorithm visits \( x, y \) before \( x \) and contracts \( x, y \) into one vertex in \( H_c \). This modified BFS does not add any extra asymptotic running time to Nbr-Kernel. Thus, the total running time is \( O(n + m) \). \( \square \)

The following lemma is used to show that Prt-Branch is FPT (Theorem 4.11).

**Lemma A.4.** Given a graph \( G = (V, E) \) and landmarks \( L \subseteq V \) with neighborhood kernel \( H = (V, E') \) and compact neighborhood kernel \( (H_c, \psi) \), \( \deg_{H_c}(v) \leq \deg_H(v) \). Also, Compact-Nbr-Kernel performs only contractions on \( H \), so \( \deg_{H_c}(v) \leq \deg_H(v) \).

**Proof.** We begin with Nbr-Kernel, where the check in Line 4 enforces Definition 4.7, ensuring correctness. For every vertex \( v \in V \), the distance from the closest landmark can be computed by BFS from \( L \) in \( O(n + m) \) time. The algorithm checks all edges in \( O(m) \), resulting in \( O(n + 2m) = O(n + m) \).

Turning to Compact-Nbr-Kernel, we traverse all vertices in BFS order from the landmarks, which guarantees each bag’s representative to be the closest to \( L \). By the definition of BNPart, Line 4 ensures that all bag members must be assigned to the same landmark because a vertex must follow its in-neighbor’s assignment. Lastly, \( \phi(v) \) is maximal for every \( v \in V \), because otherwise, there exists a vertex \( x \in V \), with at least two in-neighbors \( y, z \) such that \( v = \psi(y) = \psi(z) \neq \psi(x) \). This cannot happen because the algorithm visits \( x, y \) before \( x \) and contracts \( x, y \) into one vertex in \( H_c \). This modified BFS does not add any extra asymptotic running time to Nbr-Kernel. Thus, the total running time is \( O(n + m) \). \( \square \)

**Algorithm 3: Prt-Layer \((G, L)\)**

**Input:** Graph \( G = (V, E) \), landmarks \( L = \{v_1, \ldots, v_l\} \subseteq V \)

**Output:** Neighborhood partitioning \( \Pi \)

1. \( H \leftarrow \text{Nbr-Kernel}(G) \)
2. Partition \( V \) into layers \( \{V_i = \{v \in V : d(v, L) = i\}\} \)
3. Initialize map \( f : V \rightarrow L \) to \( f(u) = u \) for all \( u \in L \)
4. For \( i = 1 \) to \( \max_{v \in V} d(v, L) \) do
   5. \( f \leftarrow \{(f(x), x) \mid x \in \bigcup_{j=i}^{i-1} V_j\} \)
6. \( f \leftarrow f \cup \{(f(w), v) \mid v \in V_i, w \in N^r_H(v)\} \)
7. Update \( f \) with \( \text{SBAP}(L, \bigcup_{j=0}^{i-1} V_j, R) \)
8. Return \( f^{-1} \) // map from \( L \) to assigned vertices

### B EXPERIMENTAL SETUP

Datasets, along with a summary of their invariants, are available at [13]. Metagenomic cDBGs were generated by BCAIM 2 (v2.21.4) [4] with k-size 31. Some non-metagenome networks originated from the Network Data Repository [17]. For disconnected graphs, we used the largest connected component.

We ran all experiments on identical hardware, equipped with 40 CPUs (Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz) and 190 GB of memory, and running CentOS Linux release 7.9.2009. spacegraphcats is written in Python 3. We used Gurobi Optimizer 9.1.2 for ILP and QP. We used Plotly for creating charts. Our algorithms are written in C++ with OpenMP and compiled with gcc 10.2.0.

Experimental results are fully replicable using the code and data at [13]; detailed instructions in README.md.